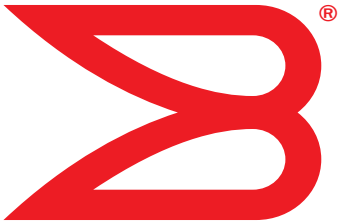
53-1002440-03
June 2012



ServerIron ADX

Security Guide

Supporting Brocade ServerIron ADX version 12.4.00a

BROCADE

Brocade Communications Systems, Incorporated

Corporate and Latin American Headquarters
Brocade Communications Systems, Inc.
130 Holger Way
San Jose, CA 95134
E-mail: info@brocade.com

Asia-Pacific Headquarters
Brocade Communications Systems China HK, Ltd.
No. 1 Guanghua Road
Chao Yang District
Units 2718 and 2818
Beijing 100020, China
Tel: +8610 6588 8888
Fax: +8610 6588 9999
E-mail: china-info@brocade.com

European Headquarters
Brocade Communications Switzerland Sàrl
Centre Swissair
Tour B - 4ème étage
29, Route de l'Aéroport
Case Postale 105
CH-1215 Genève 15
Switzerland
Tel: +41 22 799 5640
Fax: +41 22 799 5641
E-mail: emea-info@brocade.com

Asia-Pacific Headquarters
Brocade Communications Systems Co., Ltd. (Shenzhen WFOE)
Citic Plaza
No. 233 Tian He Road North
Unit 1308 – 13th Floor
Guangzhou, China
Tel: +8620 3891 2000
Fax: +8620 3891 2111
E-mail: china-info@brocade.com

Document History

Title	Publication number	Summary of changes	Date
ServerIron ADX Security Guide	*53-1002440-01*	New document	January, 2012
ServerIron ADX Security Guide	*53-1002440-02*	Corrections made to ACL chapter	April, 2012
ServerIron ADX Security Guide	*53-1002440-03*	Updates made to documentation.	June, 2012

Contents

Chapter 6 Secure Socket Layer (SSL) Acceleration

About This Document

Audience

This document is designed for system administrators with a working knowledge of Layer 2 and Layer 3 switching and routing.

If you are using a Brocade Layer 3 Switch, you should be familiar with the following protocols if applicable to your network – IP, RIP, OSPF, BGP, ISIS, IGMP, PIM, DVMRP, and VRRP.

Supported hardware and software

Although many different software and hardware configurations are tested and supported by Brocade Communications Systems, Inc. for 12.3 documenting all possible configurations and scenarios is beyond the scope of this document.

The following hardware platforms are supported by this release of this guide:

- ServerIron ADX 1000
- ServerIron ADX 4000
- ServerIron ADX 8000
- ServerIron ADX 10000

Document conventions

This section describes text formatting conventions and important notice formats used in this document.

Text formatting

The narrative-text formatting conventions that are used are as follows:

bold text	Identifies command names
	Identifies the names of user-manipulated GUI elements
	Identifies keywords
	Identifies text to enter at the GUI or CLI
italic text	Provides emphasis
	Identifies variables
	Identifies document titles
code text	Identifies CLI output

For readability, command names in the narrative portions of this guide are presented in bold: for example, **show version**.

Notes, cautions, and danger notices

The following notices and statements are used in this manual. They are listed below in order of increasing severity of potential hazards.

NOTE
A note provides a tip, guidance or advice, emphasizes important information, or provides a reference to related information.

CAUTION

A Caution statement alerts you to situations that can be potentially hazardous to you or cause damage to hardware, firmware, software, or data.

DANGER

A Danger statement indicates conditions or situations that can be potentially lethal or extremely hazardous to you. Safety labels are also attached directly to products to warn of these conditions or situations.

Notice to the reader

This document may contain references to the trademarks of the following corporations. These trademarks are the properties of their respective companies and corporations.

These references are made for informational purposes only.

Corporation	Referenced Trademarks and Products
Sun Microsystems	Solaris

Corporation	Referenced Trademarks and Products
Microsoft Corporation	Windows NT, Windows 2000
The Open Group	Linux

Related publications

The following Brocade documents supplement the information in this guide:

- *Release Notes for ServerIron Switch and Router Software TrafficWorks 12.2.00*
- *ServerIron ADX Graphical User Interface*
- *ServerIron ADX Server Load Balancing Guide*
- *ServerIron ADX Advanced Server Load Balancing Guide*
- *ServerIron ADX Global Server Load Balancing Guide*
- *ServerIron ADX Security Guide*
- *ServerIron ADX Administration Guide*
- *ServerIron ADX Switching and Routing Guide*
- *ServerIron ADX Firewall Load Balancing Guide*
- *ServerIron ADX Chassis Hardware Installation Guide*
- *Ironware MIB Reference Manual*

Getting technical help

To contact Technical Support, got to http://www.brocade.com/services-support/index.page for the latest e-mail and telephone contact information..

Chapter

Network Security 1

TCP SYN attacks

ServerIron software contains many intrusion detection and prevention capabilities. The ServerIron can be configured to defend against a variety of TCP SYN attacks, Denial of Service (DoS) attacks, and Smurf attacks.

TCP SYN attacks disrupt normal traffic flow by exploiting the way TCP connections are established.

When a normal TCP connection occurs, the connecting host first sends a TCP SYN packet to the destination host. The destination host (actually the ServerIron, acting as an intermediary between the source and destination hosts) responds with a SYN ACK packet. The connecting host then returns an ACK packet. This process, known as a "**TCP three-way handshake**", establishes the TCP connection.

A TCP SYN attack floods a host with TCP SYN packets. For each of these TCP SYN packets, the ServerIron responds with a SYN ACK packet and adds an entry to its session table. However, no ACK packet is actually sent back, so the connection is incomplete. If the attacker sends enough TCP SYN packets, the session table fills up with incomplete connections, and service can be denied to legitimate TCP connections.

syn-proxy

IP TCP syn-proxy

Configure the **ip tcp syn-proxy** command as shown in the following.

1. Configure syn-proxy in the global mode.

   ```
   ServerIronADX(config)# ip tcp syn-proxy
   ```

   Syntax: ip tcp syn-proxy

   NOTE
   You must configure ip tcp syn-proxy command only at the global level, to turn on and off the global syn-proxy flag.

2. Enable syn-proxy on each interface handling inbound SYN requests (no change here).

   ```
   ServerIronADX(config)#interface e 3/1
   ServerIronADX(config-if-3/1)# ip tcp syn-proxy in
   ```

Usage guidelines:

- The default value for a valid ACK time is 32 seconds and is not user configurable.
- If you enter a value, it is ignored. The command remains in the config file the way you enter it, in case you need to downgrade to the previous release.

- ServerIron may accept the ACK during 33 seconds to 64 seconds due to the syn-proxy algorithm, but it does not accept the ACK after 64 seconds.
- If you enter a value for the **ip tcp syn-proxy** <*value*> command from the CLI or upgrade from an older release such as 09.4.x to 09.5.2a with the **ip tcp syn-proxy** <*value*> command in the config file, you receive the following warning message.

```
Warning: The value 10 is being ignored.
         Default ACK validate time of 32 seconds will be used.
         To change the MSL value, issue 'server msl <value>'.
```

Granular application of syn-proxy feature

This feature applies to ServerIron ADX Syn-Proxy. When this feature is enabled, traffic destined to a virtual server IP is denied if the destination port is not defined under any of the virtual server definitions.

This feature prevents ServerIron ADX from responding with TCP SYN-ACK to TCP SYN for ports not defined under VIP.

Use the following command to validate traffic against a configured virtual port.

```
ServerIronADX(config)# server syn-cookie-check-vport
```

Syntax: [no] server syn-cookie-check-vport

Syn-def

Introduction

Use SYN-def (also known as SYN-Defense) to protect the hosts behind the ServerIron (not the ServerIron itself) by the ServerIron to complete the TCP three-way handshake on behalf of a connecting client. There is no SYN-cookie functionality with SYN-def.

NOTE
SYN-Defense is recommened for only where Direct Server Return (DSR) is used. DSR is not supported with SYN-proxy and is supported with SYN-def. For non DSR scenarios, use Syn-Proxy only.

show server traffic

Use the **show server traffic** command to display information about the number of times the incomplete connection threshold was reached.

```
ServerIronADX# show server traffic
Client->Server       =          0  Server->Client       =          0
Drops                =          0  Aged                 =          0
Fw_drops             =          0  Rev_drops            =          0
FIN_or_RST           =          0  old-conn             =          0
Disable_drop         =          0  Exceed_drop          =          0
Stale_drop           =          0  Unsuccessful         =          0
TCP SYN-DEF RST      =          0  Server Resets        =          0
Out of Memory        =          0  Out of Memory        =          0
```

The last line contains information relevant to the incomplete connection threshold. The TCP SYN-DEF RST field displays the number of times the incomplete connection threshold was reached. The Server Resets field displays the number of times the ServerIron sent a TCP RESET packet to the destination real server.

SYN-def-dont-send-ack

The SYN-def feature allows the ServerIron to complete the TCP three-way handshake on behalf of a connecting client. When a connecting client sends a TCP SYN to a server, the ServerIron forwards the SYN to the real server, then forwards the SYN ACK from the server to the client. Next, the ServerIron sends an ACK to the real server, completing the three-way handshake on behalf of the connecting client. This action allows the real server to move the connection from its pending connection queue to its established (and much larger) connection queue.

Use the **server syn-def-dont-send-ack** command to prevent the ServerIron from sending the ACK to the real server to complete the three-way handshake.

Example

```
ServerIronADX(config)#server syn-def-dont-send-ack
```

show server debug

Use the **show server debug** command to display information about the configuration, as shown in the following example.

```
SLB-chassis1/1#show server debug


 Generic Deug Info
BP Distribution      =          Enabled  JetCore              =              No
No of BPs            =                3  No of Partner BPs    =               0
Partner Chassis MAC  =  0000.0000.0000
Partner BP1 MAC      =  0000.0000.0000  Partner BP2 MAC      =  0000.0000.0000
Partner BP3 MAC      =  0000.0000.0000  Partner BP4 MAC      =  0000.0000.0000
Partner BP5 MAC      =  0000.0000.0000  Partner BP6 MAC      =  0000.0000.0000

 Server Load Balancing Debug Info
Total Get            =           3  Total Free           =          0
Get Fails            =           0  Get Buffer failure   =          0
Forward Sp           =           0  Reverse Sp           =          0
Bad creates          =           0  TCP Resets           =          0
Fw resets            =           0  Rev Resets           =          0
Double Free          =           0  Error                =          0
Free inv Sess Idx    =           0  Free list Idx inv    =          0
Cache-Reassigns      =           0  Trans-Denied         =          0
Multi Path Fwd Use   =           0  Multi Path Rev Use   =          0
Bad non-owner        =           0  Select Fwall         =          0
FTP-trans-error      =           0  Cache track-error    =          0
Fw tcp inside move   =           0  Fw udp inside move   =          0
Fw SYNC delayed      =           0  ownership contention =          0
FW stale to conns    =           0  FW stale to delq con =          0
FW stale from conns  =           0  FW stale from delq c =          0
FW stale from nuke c =           0  Sac frwds            =          0

Unxpectd udata       =           0  Unxpectd udata(def)  =          0
Client->Server       =           0  Server->Client       =          0
Drops                =           0  Aged                 =          0
Fw_drops             =           0  Rev_drops            =          0
FIN_or_RST           =           0  old-conn             =          0
Disable_drop         =           0  Exceed_drop          =          0
Stale_drop           =           0  Unsuccessful         =          0
SYN def/proxy RST    =           0  Server Resets        =          0
Out of Memory        =           0  Out of Memory        =          0
last conn rate       =           0  max conn rate        =          0
last TCP attack rate =           0  max TCP attack rate  =          0
fast vport found     =           0  fast vport n found   =          0
Fwd to non-static FI =           0  Dup stale SYN        =          0

TCP forward FIN      =           0  TCP reverse FIN      =          0
Fast path FWD FIN    =           0  Fast path REV FIN    =          0
Fast path SLB SYN    =           0  Dup SYN after FIN    =          0
Duplicate SYN        =           0  Duplicate sessions   =          0
TCP ttl FIN recvd    =           0  TCP ttl reset recvd  =          0
Sessions in DEL_Q    =           0  Sess force deleted   =          0
Fwd sess not found   =           0  sess already in delQ =          0
Sess rmvd from delQ  =           0
Fragment buf full er =           0  Incoming TCP cksum e =          0
New sess sync sent   =           0  New sess sync recvd  =          0
L4 msg sent          =           0  L4 msg recvd         =          0
foundry packet sent  =           0  ipc packet sent      =    2818942
TCP SYN received     =           0  TCP SYN dropped      =          0
TCP SYN to MP        =           0  TCP SYN ACK to MP    =          0
TCP SYN ACK received =           0  TCP SYN ACK dropped  =          0
TCP pkt received     =           0  TCP pkt dropped      =          0
TCP pkt to MP        =           0 PBSLB tftp status    = In progres

Avail. Sessions      =     1999996  Total Sessions       =    2000000
Hash size            =      200001

Total C->S Conn      =           0  Total S->C Conn      =          0
Total Reassign       =           0  Unsuccessful Conn    =          0
Server State - 0: diasbled, 1:enabled, 2:failed, 3:test, 4:suspect, 5:grace_dn, 6:active

Real Server   St CurrConn          TotConn    TotRevConn CurrSess   PeakConn

R1            1  0/0/0             0          0          0          0
rs1           1  0/0/0             0          0          0          0
```

No response to non-SYN first packet of a TCP flow

ServerIron can remain passive for non-SYN packet in the beginning of the flow. The default behavior is to send TCP RESET to client when a non-SYN packet is received in the beginning.

By default, when ServerIron ADX receives TCP packet that is destined to VIP and there is no session match then it sends TCP reset to the sender. However, if one desires to remain passive then the above feature can be enabled.

To not send the reset packet, use the following command.

```
ServerIronADX(config)# server reset-on-syn-only
```

To remove the configuration, use the following command.

```
ServerIronADX(config)# no server reset-on-syn-only
```

Syntax: [no] server reset-on-syn-only

Prioritizing management traffic

ServerIron ADX software allows the system to prioritize traffic destined to the management IP address in order to facilitate uninterrupted access to the ServerIron switch even under heavy load conditions. This feature allows you to prioritize management traffic based on the following.

1. Client IP address/subnet
2. Protocol (TCP/UDP/IP) and
3. TCP or UDP port number

With this feature turned on, the specified traffic is directly forwarded to the Management Module in hardware. In the following example, traffic from the source subnet 1.1.1.1 and destined to management IP 10.45.16.104 for TCP port 22 (SSH) is prioritized.

```
ServerIronADX(config)# server prioritize-mgmt-traffic 1.1.1.1 255.255.255.0
10.45.16.104 6 22
```

Syntax: server prioritize-mgmt-traffic <*source ip*> <*mask*> <*destination ip*> [<*protocol*>] [<*port*>]

The <*source ip*> variable specifies the Source IP address.

The <*mask*> variable specifies the Mask for the source IP address.

The <*destination ip*> variable specifies the Destination management IP address. The destination IP address must already be configured on the ServerIron ADX. If the IP address is not configured, the command is rejected.

The <*protocol*> variable specifies any protocol.

The <*port*> variable specifies a TCP or UDP port.

It is also possible to prioritize management traffic from any source ip as shown in the example below.

```
ServerIronADX(config)# server prioritize-mgmt-traffic any 10.45.16.104 6 22
```

Syntax: [no] server prioritize-mgmt-traffic any <*destination ip*> [<*protocol*>] [<*port*>]

NOTE
The prioritizing management traffic feature should not be enabled for a ServerIron ADX router VE address if this interface is used for source-NAT as that would break the SLB traffic flow.

Refer to the following examples.

Prioritization of TCP port 80 traffic to management IP 200.1.1.1

```
ServerIronADX# server prioritize-mgmt-traffic 1.1.1.1 255.255.255.0 200.1.1.1 6
80
```

Prioritization of TCP port 80 traffic to management IP 200.1.1.1 from any source IP address

```
ServerIronADX# server prioritize-mgmt-traffic any 200.1.1.1 6 80
```

Prioritization of UDP port 2222 traffic to management IP 200.1.1.1

```
ServerIronADX# server prioritize-mgmt-traffic 1.1.1.1 255.255.255.0 200.1.1.1 17
2222
```

Prioritization of IP protocol 89 (OSPF) traffic to management IP 200.1.1.1

```
ServerIronADX# server prioritize-mgmt-traffic 1.1.1.1 255.255.255.0 200.1.1.1 89
```

Protection against attack in hardware

ServerIron ADX allows for protection against attack in hardware without impacting MP or BP CPU utilization. Configure the **server the drop-all-mgmt-access** command to drop all traffic destined to a specified management IP address.

The following command drops all traffic destined to the management IP address 10.45.16.104.

```
ServerIronADX(config)# server drop-all-mgmt-access 10.45.16.104
```

Syntax: [no] server drop-all-mgmt-access <*destination ip*>

NOTE
For a router, the destination IP address is the physical or ve interface IP address For a switch, the destination IP address is the management IP address.

The server drop-all-mgmt-access feature when used in combination with the server prioritize-mgmt-traffic feature allows you to prioritize valid traffic while blocking unwanted traffic destined to the management IP address.

For example, with the following configuration, only ssh, telnet and http traffic destined to management IP address 10.45.16.104 will be prioritized and all other traffic destined to 10.45.16.104 will be dropped.

```
ServerIronADX(config)#server prioritize-mgmt-traffic 1.1.1.1 255.255.255.0
10.45.16.104 6 22
ServerIronADX(config)#server prioritize-mgmt-traffic 1.1.1.1 255.255.255.0
10.45.16.104 6 23
ServerIronADX(config)#server prioritize-mgmt-traffic 1.1.1.1 255.255.255.0
10.45.16.104 6 80
ServerIronADX(config)#server drop-all-mgmt-access 10.45.16.104
```

Peak BP utilization with TRAP

Show CPU-utilization command enhancement

The **show cpu-utilization** command displays CPU utilization peaks since the system boot or the last reset of counters (using the **clear cpu utilization** command).

The command, **clear cpu-utilization**, on both the MP and the BP is used to reset the counter.

BP utilization threshold

The **bp-utilization-threshold** command allows you to specify a threshold for BP CPU utilization. Define this command under the global configuration mode.

When the threshold is exceeded, the event is logged and a trap is sent. The log and trap are rate-limited to one per two minutes.

The command takes a percentage string as parameter.

Example

```
ServerIronADX(config)# bp-utilization-threshold 80.5%
```

Syntax: bp-utilization-threshold <*percentage*>

MP utilization threshold

The **mp-utilization-threshold** command specifies a threshold for BP CPU utilization. Define this command under the global configuration mode.

When the threshold is exceeded, the event is logged and a trap is sent. The log and trap are rate-limited to one every two minutes.

The command takes a percentage string as parameter.

Example

```
ServerIronADX(config)# mp-utilization-threshold 80.5%
```

Syntax: mp-utilization-threshold <*percentage*>

Transaction Rate Limit (TRL)

Transaction Rate Limit, allows the ServerIron ADX to monitor and limit traffic from any one IP address.

Understanding transaction rate limit

Transaction Rate Limit counts the number of transactions received from any one IP address. If the transaction count exceeds a specified threshold value, traffic from that IP address is held and not processed for a specified number of minutes.

Transaction rate limit provides the flexibility to specify different configurations for different clients, based on the client IP address/prefix.

Transaction rate limit provides the following benefits:

- Ability to apply a default transaction rate limit value to all clients, while maintaining an exception list.
- Ability to apply a different transaction rate limit rate per client IP or prefix.
- Ability to exclude specific IP addresses or prefixes from transaction rate limit and maintain an exclude list.
- Ability to apply transaction rate limit to traffic coming to a specific VIP only.

- Ability to operate on a per VIP basis, whereby a different rate limit can be applied to traffic coming to a different VIP.

Configuring transaction rate limit

To enable transaction rate limit, you must configure parameters for each client address/prefix and apply the transaction rate limit configuration to a specific VIP.

Prerequisites

Before you can configure transaction rate limit, you must configure a virtual server. The following example shows how to configure a virtual server.

```
ServerIronADX> enable
ServerIronADX# config terminal
ServerIronADX(config)# server virtual-name-or-ip bwVIP 1.1.1.33
```

Syntax: [no] server virtual-name-or-ip <*vip-name-or-address*> <*ip address*>

Configure transaction rate limit rule set

The transaction rate limit parameters are grouped into a set and each set is associated with a name. To create a set of transaction rate limit rules, follow these steps.

1. Enable privileged EXEC mode.

   ```
   ServerIronADX> enable
   ```

2. Enter global configuration mode.

   ```
   ServerIronADX# configure terminal
   ```

3. Configure name of a transaction rate limit rule set and enter client transaction rate limit configuration mode.

   ```
   ServerIronADX(config)#client-trans-rate-limit tcp TRL1
   ```

   Syntax: [no] client-trans-rate-limit tcp | udp | icmp <*name*>

4. Specify the **trl** keyword for client subnet and set connection rate.

   For IPv4:

   ```
   ServerIronADX(config-client-trl-trl1)# trl 100.1.1.0 255.255.255.0
   monitor-interval 3 conn-rate 10 hold-down-time 1
   ```

   For IPv6:

   ```
   ServerIronADX(config-client-trl-trl1)# trl 100::1/128 monitor-interval 3
   conn-rate 10 hold-down-time 1
   ```

   Syntax: [no] trl { <*client-IPv4*> <*client-mask*> | <*client-IPv6*> <*prefix*> **} monitor-interval** <*mon-value*> **conn-rate** <*con-value*> **hold-down-time** <*hold-down-value*>

Configure transaction rate limit to exclude a client

You can configure a client address/prefix to be excluded from transaction rate limiting within a transaction rate limit configuration group.

To exclude a client from transaction rate limit, follow these steps.

1. Enable privileged EXEC mode.

   ```
   ServerIronADX> enable
   ```

2. Enter global configuration mode.

   ```
   ServerIronADX# configure terminal
   ```

3. Specify the name of the transaction rate limit rule set and enter client transaction rate limit configuration mode.

   ```
   ServerIronADX(config)# client-trans-rate-limit tcp TRL1
   ```

   Syntax: [no] client-trans-rate-limit tcp | udp | icmp <*name*>

4. Specify the **trl** parameter for the client subnet and the **exclude** keyword.

   For IPv4:

   ```
   ServerIronADX(config-client-trl-TRL1)# trl 100.1.1.0 255.255.255.0 exclude
   ```

   For IPv6:

   ```
   ServerIronADX(config-client-trl-TRL1)# trl 300::1/128 exclude
   ```

   Syntax: [no] trl { <*client-IPv4*> <*client-mask*> | <*client-IPv6*> <*prefix*> **} exclude**

Configure a transaction rate limit default

You can specify a default transaction rate limit configuration for all other clients that are not explicitly configured. To create a transaction rate limit default for a group, follow these steps.

1. Enable privileged EXEC mode.

   ```
   ServerIronADX> enable
   ```

2. Enter global configuration mode.

   ```
   ServerIronADX# configure terminal
   ```

3. Specify name of transaction rate limit rule set and enter client transaction rate limit configuration mode.

   ```
   ServerIronADX(config)# client-trans-rate-limit tcp TRL1
   ```

   Syntax: [no] client-trans-rate-limit tcp | udp | icmp <*name*>

4. Specify the default trl parameter for this group.

   ```
   ServerIronADX(config-client-trl)# trl default monitor-interval 3 conn-rate 10
   hold-down-time 1
   ```

   Syntax: [no] trl default monitor-interval <*mon-value*> **conn-rate** <*con-value*> **hold-down-time** <*hold-down-value*>

Configure transaction rate limit for pass through traffic

You can configure transaction rate limit for traffic that is not going to a virtual server. You can configure only one group for pass through traffic.

To create a transaction rate limit group for pass through traffic, follow these steps.

1. Enable privileged EXEC mode.

```
ServerIronADX> enable
```

2. Enter global configuration mode.

```
ServerIronADX# configure terminal
```

3. Specify name of BW rule set and enter client bandwidth configuration mode.

```
ServerIronADX(config)# client-trans-rate-limit tcp default
```

Syntax: [no] client-trans-rate-limit tcp | udp | icmp default

4. Specify the trl parameter for the client subnet and set a connection rate.

For IPv4:

```
ServerIronADX(config-client-trl)#trl 100.1.1.0 255.255.255.0 monitor-interval
3 conn-rate 10 hold-down-time 1
```

For IPv6:

```
ServerIronADX(config-client-trl)#trl 300:11/128 monitor-interval 3 conn-rate
10 hold-down-time 1
```

Syntax: [no] trl { <*client-IPv4*> <*client-mask*> | <*client-IPv6*> <*prefix*> **} monitor-interval** <*mon-value*> **conn-rate** <*con-value*> **hold-down-time** <*hold-down-value*>

5. The transaction rate limit policy pertaining to the protocol and the port must be applied to either the physical or the virtual interface for pass through traffic. This will ensure that the traffic is brought to the application processor (BP) for rate-limitation.

Applying policy on physical interface

```
ServerIronADX(config) # interface eth 1/1
ServerIronADX(config-if-1/1) # ip tcp trans-rate 80
```

Applying policy on virtual interface

```
ServerIronADX(config) # interface ve 20
ServerIronADX(config-vif-20) # ip udp trans-rate 53
```

Syntax: [no} ip tcp | udp trans-rate <ports>

Syntax: [no} ip icmp trans-rate

The <ports> parameter specifies one or more TCP or UDP ports to monitor. You can monitor up to four ports.

Apply transaction rate limit to a VIP

After configuring transaction rate limit, you must bind transaction rate limit to a VIP. To enable transaction rate limit, follow these steps.

1. Enable privileged EXEC mode.

```
ServerIronADX> enable
```

2. Enter global configuration mode.

```
ServerIronADX# configure terminal
```

3. Specify **server virtual-name-or-ip** command and VIP name to enter virtual server configuration mode.

```
ServerIronADX(config)# server virtual-name-or-ip bwVIP
```

Syntax: [no] server virtual-name-or-ip <*name-or-address*>

4. Specify the BW parameter and BW rule set.

```
ServerIronADX(config-vs-bwVIP)# client-trans-rate-limit trl
```

Syntax: [no] client-trans-rate-limit <*name*>

5. The transaction rate limit policy pertaining to the protocol and the port must be applied to either the physical or the virtual interface for traffic hitting to Virtual IP.

Applying policy on physical interface

```
ServerIronADX(config) # interface eth 1/1
ServerIronADX(config-if-1/1) # ip tcp trans-rate 80
```

Applying policy on virtual interface

```
ServerIronADX(config) # interface ve 20
ServerIronADX(config-vif-20) # ip udp trans-rate 53
```

Syntax: [no} ip tcp | udp trans-rate <ports>

Syntax: [no} ip icmp trans-rate

The <ports> parameter specifies one or more TCP or UDP ports to monitor. You can monitor up to four ports.

Deleting all TRL rules in a policy

You can delete all TRL rules in a policy as shown.

```
ServerIronADX(config)# client-trans-rate-limit tcp trl1
ServerIronADX(config-client-trl-trl1)# trl delete-all-rules
```

Syntax: trl delete-all-rules

Download transaction rate limit configuration from a TFTP server. (optional)

When a Transaction Rate Limit configuration becomes very large, you can download the configuration from a TFTP server.

NOTE
A TRL configuration file can have IPv4 as well as IPv6 rules.

The following example shows how to download a Transaction Rate Limit configuration from a TFTP server.

```
ServerIronADX(config)# server trl tftp 100.1.1.1 test.trl 2
```

Syntax: server trl tftp <*ip-address*> <*trl_config_file_name*> <*retry_count*>

Specify the following values.

<*ip_address*> —IP address of the TFTP server.

<*trl_config_file_name*> —File name of Transaction Rate Limit configuration.

<*retry_count*> —Retry number for the download.

Verify that the Transaction Rate Limit configuration file is in the following format.

```
client-trans-rate-limit tcp trl101
trl 10.2.24.0/24 monitor-interval 50 conn-rate 100 hold-down-time 60
trl 10.2.24.10/32 exclude
```

NOTE
This is the same format as the **show running-configuration** command generates.

Configuring the maximum number of rules

By default a TRL a policy can have up to 2500 IPv4 rules and 2500 IPv6 rules. A maximum of 15,000 IPv4 and 15,000 IPv6 rules are supported on a ServerIron ADX for all policies. While the maximum number of rules cannot be increased over the 15,000 maximum, these limits can be changed globally or locally per-policy.

Changing the maximum number of rules globally.

You can change the maximum number of TRL rules globally on a ServerIron ADX for all policies as shown.

```
ServerIronADX(config)# client-trans-rate-limit max-ipv4-rules 2000
```

Syntax: [no] client-trans-rate-limit { max-ipv4-rules | max-ipv6-rules } <*rules-count*>

The **max-ipv4-rules** parameter specifies that the rules limit is being set for IPv4 rules.

The **max-ipv6-rules** parameter specifies that the rules limit is being set for IPv6 rules.

The <*rules-count*> variable specifies the number of rules that will be supported globally. The maximum values (also the default) are: 15,000 for IPv4 and 15,000 for IPv6.

Changing the maximum number of rules locally per-policy.

You can change the maximum number of TRL rules for an individual policy on a ServerIron ADX for as shown.

```
ServerIronADX(config)# client-trans-rate-limit tcp trl1
ServerIronADX(config-client-trl-trl1)# trl max-ipv4-rules 2000
```

Syntax: [no] trl { max-ipv4-rules | max-ipv6-rules } <*rules-count*>

The **max-ipv4-rules** parameter specifies that the rules limit is being set for IPv4 rules for the specified policy.

The **max-ipv6-rules** parameter specifies that the rules limit is being set for IPv6 rules for the specified policy.

The <*rules-count*> variable specifies the number of rules that will be supported for the specified policy that this command is being configured under. The default values are: 2500 for IPv4 and 2500 for IPv6. The value for each (IPv4 and IPv6) can be set to any number as long as the global limits are observed.

Saving a TRL configuration

The following applies to saving a TRL config:

- the startup-config cannot store 15,000 IPv4 and 15,000 IPv6 rules.
- If the total number of IPv4 and IPv6 rules exceeds 2500, issuing the **write mem** command stores the TRL rules in the "trl_conf.txt" file on the internal USB drive.
- the policy config and global/local maximum rule count config is always stored in the startup-config.

Disabling the storage of TRL rules on the internal USB drive

By default, storage of TRL rules on the internal USB drive of a ServerIron ADX is enabled. You can disable the storage of TRL rules on the internal USB drive of a ServerIron ADX as shown.

```
ServerIronADX(config)# no client-trans-rate-limit usb-config-gen
```

Syntax: no client-trans-rate-limit usb-config-gen

NOTE
Where the storage of TRL rules on the internal USB drive of a ServerIron ADX is disabled and the total rules exceeds 2500, only 2500 rules would be saved in startup-config.

Transaction rate limit command reference

This section describes the syntax, semantics, and usage for each transaction rate limit command. This section contains the following sections:

- "client-trans-rate-limit"
- "trl"

client-trans-rate-limit

Use the **client-trans-rate-limit** command in the global configuration mode to configure a transaction rate limit rule name and traffic type.

Syntax: client-trans-rate-limit {icmp <*name*> **| default} | {tcp** <*name*> **| default} | {udp** <*name*> **| default}**

icmp - Specifies ICMP transaction rate limit for client subnet.

tcp - Specifies TCP transaction rate limit for client subnet.

udp - Specifies UDP transaction rate limit for client subnet.

<*name*> - Specifies the name for this configuration.

default - Specifies default.

trl

Use the **trl** command in the global configuration client-trl mode to configure transaction rate limit rules.

Syntax: **trl {default | {** <*client-IPv4*> <*client-mask*> | <*client-IPv6*> <*prefix*> **} {exclude | monitor-interval** <*monitor-value*> **conn-rate** <*connection-value*> **hold-down-time** <*hold-down-value*>**}}**

default - Specifies default transaction rate limit parameter.

<*client-IPv4*> - Specifies IPv4 client subnet and <*client-mask*> - Specifies the IPv4 client mask.

<*client-IPv6*> - Specifies IPv6 client subnet and <*prefix*> - Specifies the IPv6 client mask bits.

exclude - Specifies to exclude the prefix from transaction rate limit.

monitor-interval - Specifies time interval for monitoring in 100ms.

<*monitor-value*> - Specifies value of time interval for monitoring.

conn-rate - Specifies connection rate.

<*connection-value*> - Specifies value of connection rate for client.

hold-down-time - Specifies time for holding down source.

<*hold-down-value*> - Specifies hold down time in minutes.

Command modes

Global configuration mode.

Global TRL

If TRL per client subnet is not needed, Global TRL can be used to create a configuration to apply to all the incoming traffic.

Use **ip [tcp | udp | icmp] trans-rate** to enable TRL on the ServerIron for TCP, UDP, or ICMP traffic. If any more than a specified number packets per second come from the same IP address over a specified interval, then all traffic from that IP address is held down for a specified number of minutes.

Syntax: **[no] ip [tcp | udp | icmp] trans-rate monitor-interval** <*interval*> **conn-rate** <*rate*> **hold-down-time** <*minutes*>

monitor-interval <*interval*> Amount of time used to measure incoming traffic. This parameter is specified in increments of 100ms. For example, to measure traffic over a 1 second interval, you would specify 10 for this.

conn-rate <*rate*> Threshold for the number of connections per second from any one IP address. Traffic exceeding this rate over the specified interval is subject to hold down.

hold-down-time <*minutes*> Number of minutes that traffic from an IP address that has sent packets at rate higher than the configured threshold is to be held down.

Example

```
ServerIronADX(config)# ip tcp trans-rate monitor-interval 600 conn-rate 100
hold-down-time 5
```

This command configures the ServerIron to monitor incoming TCP traffic. If more than 100 TCP connections per second arrive from the same IP address over a 60-second interval (600 X 100ms), then all TCP traffic from that IP address is held down for 5 minutes.

To apply TRL to TCP traffic coming into port 80 on interface 1/1.

```
ServerIronADX(config)# interface ethernet 1/1
ServerIronADX(config-if-1/1)# ip tcp trans-rate 80
```

where <*ports*> sets one or more TCP or UDP ports to monitor. With TRL, the ServerIron can monitor up to 4 specific ports. The ServerIron can also monitor traffic to all the ports by configuring the default port.

TRL plus security ACL-ID

Even though TRL is applied to an interface and effects all traffic received on this interface, with the **security acl-id** <*acl-num*> command TRL can be applied only to specific traffic coming in on that interface.Refer to "security acl-id" on page 15.

security acl-id

The **security** global command accepts **acl-id** <*acl-num*> as a parameter.

Syntax: [no] security acl-id <*id*>

Example

```
ServerIronADX(config)# security acl-id 4
```

Once **security acl-id** <*acl-num*> is configured, only packets matching the configured ACL will be subject to the L4 security rules configured on the system. (Specifically, TRL and manual hold down will take effect only for packets matching this configured ACL). If you want specific traffic to bypass the L4 security features, then do not include those IP addresses in the access list.

NOTE
The **security acl-id** takes precedence over all TRL configuration.

Transaction rate limit hold-down value

if you configure "**hold down 0**," the incoming request is not held down. Instead it generates a log.

Displaying TRL rules statistics

You can display statistics for TRL rules as shown.

```
ServerIronADX#show client-trl rules-stat
Policy-Name default-rule ipv4-rules-alloted ipv4-rules-added ipv6-rules-alloted ipv6-rules-added
trl1        0            2500               0                2500               0
trl2        0            2500               0                2500               0
trl3        0            2500               0                2500               0
Global ipv4 rule num: 2500, total-alloted-ipv4-rules: 7500
Global ipv6 rule num: 2500, total-alloted-ipv6-rules: 7500
```

Syntax: show client-trl rules-stat

Displaying TRL rules in a policy

You can display TRL rules in a policy as shown.

```
ServerIronADX#show client-trl trl-policy1 ipv6 40
Max Count: 2500    Total Count: 2

IP address/Mask            interval    attempts    holddown
---------------            --------    --------    --------
300::3a95/128              1           67          93
300::3a96/128              66          38          34
```

Syntax: show client-trl <*policy-name*> **{ ipv4 | ipv6}** <*index*>

The <*policy-name*> variable specifies the TRL policy that you want to display rules for.

The **show client-trl** command displays entries in the TRL policy list, starting from the point specified with the <*index*> parameter.

Displaying IP address with held down traffic

To display a list of IPv4 and IPv6 addresses whose traffic has been held down, enter commands such as the following.

```
ServerIronADX# rconsole 2 1
ServerIronADX2/1 #show security holddown

source         destination    vers attempt start    last     HD time
192.168.2.30   Any tcp              0     000ab6ae  00000000 Y  9
192.168.2.40   Any tcp              0     000ab6ea  00000000 Y  9
```

Syntax: rconsole <*slotnum*> <*cpunum*>

Syntax: show security holddown

The following table lists the output from the **show security holddown** command.

TABLE 1 Output from the show security holddown command

Field	Description
source	Source IPv4 or IPv6 address that is currently being held down
destination	TCP, UDP, or ICMP depending on the type of traffic sent by the client.
vers	Used by Brocade Technical Support.
attempt	Number of connection attempts made by the client during the current monitoring interval.
start	Time stamp representing the start of the monitoring interval.
last	Time stamp representing the last time the ServerIron received a connection request from the client.
HD	Whether the IP address is currently being held down. Y indicates that the address is being held down. N indicates that it is not.
time	Time remaining for this IP address to be held down, if the HD field contains Y.

Refusing new connections from a specified IP address

Use the **security hold-source-ip** command to refuse new connections from a specified IP address for a specified amount of time. This feature applies to all TCP, UDP, and ICMP traffic originating from the specified IP address.

Syntax: [no] security hold-source-ip <*ip-address*> <*minutes*>

Example

To configure the ServerIron to refuse connections from 192.168.9.210 for 20 minutes, enter.

```
ServerIronADX(config)# security hold-source-ip 192.168.9.210 20
```

To display the IP addresses from which connections are currently being refused.

```
ServerIronADX# rconsole 2 1
ServerIronADX2/1 # show security holddown

source         destination    vers attempt start    last     HD time
192.168.2.30   Any tcp             0     000ab6ae   00000000 Y  9
192.168.2.40   Any tcp             0     000ab6ea   00000000 Y  9
```

The IP addresses for which connections are being refused are displayed in the source column.

HTTP TRL

This section describes how to use the HTTP Transaction Rate Limiting (TRL) feature with ServerIron devices.

Overview of HTTP TRL

HTTP TRL provides HTTP transaction rate limiting for SSL and HTTP traffic, based on a customer ID. Existing ServerIron TRL features, which are based on source IP addresses, are inadequate in environments where a client is identified by an application user ID. HTTP TRL allows you to prevent per-client over subscription by allowing you to configure features, such as transaction and connection rate limiting, based on customer IDs.

With HTTP TRL, the rate limit configuration for each customer is grouped into a set. Each of these groups can be applied to multiple VIPs. A counter is maintained on per-VIP basis. When a client request is received, the client customer ID is extracted and decoded. A table lookup is performed on the customer ID and, if the client is subjected to a rate limit, a session lookup is done to locate the current connection information.

For each BP, the current counter is checked against the configuration. If the limit is exceeded, the configured action occurs.

HTTP TRL features

Before you configure HTTP TRL, you should be aware of the following benefits and restrictions for this feature:

- The customer ID is contained within the HTTP header, is alphanumeric, and can be up to 101 characters in length.
- Maximum customer ID entries is 35K.
- Customer ID entries can be manually configured or have dynamic upload support.
- All customer connections are supported on a single VIP with support for up to 10K connections.
- Customer report response times can run up to 120 seconds before they timeout at the gateway tier.

- Rate-limiting functionality must support rate over time and total connections, based on customer ID.
- Max-conn currently works only for HTTP1.0.
- This feature supports http redirect, or drop client response actions once rate-limit has been exceeded.
- This feature provides event and threshold alert monitoring and notification, based on specific customer connection SLAs.

Configuring HTTP TRL

This section describes how to configure the HTTP TRL feature.

NOTE
For traffic going through a VIP, Brocade recommends that you apply the TRL policy to the VIP and Interface.

Configuring HTTP TRL client

Use the following procedures to configure the HTTP TRL client rate limit and the client maximum connection.

Configuring HTTP TRL client rate limit

To configure the HTTP TRL client rate limit, follow these steps.

1. Define an HTTP TRL policy.

   ```
   ServerIronADX(config)# http-trl-policy p1
   ```

   Syntax: [no] http-trl-policy <*policy-name*>

2. Configure an HTTP TRL client rate limit.

   ```
   ServerIronADX(config-http-trl-p1)# client-name c1 monitor-interval 1 10 20 0
   ```

   Syntax: [no] client-name <*client-name*> **monitor-interval** <*interval-value*> <*warning-rate*> <*shutdown-rate*> <*holddown-interval*>

   For more detailed command information, refer to "Client-name <client-name> monitor-interval" on page 27.

3. Configure the action to take if a client exceeds the configured rate limit (optional).

   ```
   ServerIronADX(config-http-trl-p1)# client-name c1 exceed-action reset
   ```

   Syntax: [no] client-name <*client-name*> **exceed-action reset**

Configuring HTTP TRL client maximum connection

To configure HTTP TRL client maximum connection, follow these steps.

1. Define an HTTP TRL policy.

   ```
   ServerIronADX(config)# http-trl-policy p1
   ```

Syntax: [no] http-trl-policy <*policy-name*>

2. Configure an HTTP TRL client maximum connection.

```
ServerIronADX(config-http-trl-p1)# client-name c1 max-conn 10
```

Syntax: [no] client-name <*client-name*> **max-conn** <*max-conn-value*>

<*max-conn-value*>—specifies maximum number of connection client can setup.

3. Configure the action to take if a client exceeds the configured maximum connections (optional).

```
ServerIronADX(config-http-trl-p1)# client-name c1 exceed-action reset
```

Syntax: [no] client-name <*client-name*> **exceed-action reset**

Configuring HTTP TRL defaults

Use the following procedures to configure the HTTP TRL default rate limit and the default maximum connection.

Configuring HTTP TRL default rate limit

To configure HTTP TRL default rate limit, follow these steps.

1. Define an HTTP TRL policy.

```
ServerIronADX(config)# http-trl-policy p1
```

Syntax: [no] http-trl-policy <*policy-name*>

2. Configure an HTTP TRL default rate limit.

```
ServerIronADX(config-http-trl-p1)# default monitor-interval 1 10 20 0
```

Syntax: [no] default monitor-interval <*interval-value*> <*warning-rate*> <*shutdown-rate*> <*holddown-interval*>

3. Configure the action to take if a client exceeds the configured rate limit (optional).

```
ServerIronADX(config-http-trl-p1)# default exceed-action reset
```

Syntax: [no] default exceed-action reset

Configuring HTTP TRL default maximum connection

To configure HTTP TRL default maximum connection, follow these steps.

1. Define an HTTP TRL policy.

```
ServerIronADX(config)# http-trl-policy p1
```

Syntax: [no] http-trl-policy <*policy-name*>

2. Configure an HTTP TRL default maximum connection.

```
ServerIronADX(config-http-trl-p1)# default max-conn 10
```

Syntax: [no] default max-conn <*max-conn-value*>

3. Configure the action to take if a client exceeds the configured maximum connection (optional).

```
ServerIronADX(config-http-trl-p1)# default exceed-action reset
```

Syntax: [no] default exceed-action reset

Sample HTTP TRL configuration

This section describes how to configure a sample HTTP TRL configuration. This scenario describes all the required steps for configuring HTTP TRL, with notes the optional steps. This configuration consists of four parts:

- Creating an HTTP TRL policy with a client rate limit
- Configuring Layer 4 server load balancing
- Creating a CSW rule and policy with HTTP TRL
- Enabling Layer 7 server load balancing

Creating an HTTP TRL policy with client rate limit

To configure a HTTP TRL policy with client rate limit, follow these steps.

1. Define an HTTP TRL policy.

   ```
   ServerIronADX(config)# http-trl-policy p1
   ```

   Syntax: [no] http-trl-policy <*policy-name*>

2. Configure an HTTP TRL client rate limit.

   ```
   ServerIronADX(config-http-trl-p1)# client-name c1 monitor-interval 1 10 20 0
   ```

   Syntax: [no] client-name <*client-name*> **monitor-interval** <*interval-value*> <*warning-rate*> <*shutdown-rate*> <*holddown-interval*>

3. Configure the action to take if a client exceeds the configured rate limit (optional).

   ```
   ServerIronADX(config-http-trl-p1)# client-name c1 exceed-action reset
   ```

   Syntax: [no] client-name <*client-name*> **exceed-action reset**

Configuring Layer 4 SLB

To configure Layer 4 SLB, follow these steps.

1. Define a real server (1) with an IP address.

   ```
   ServerIronADX(config)# server real web1 1.1.1.1
   ```

   Syntax: server real <*real-server*> <*ip-address*>

2. Define a real HTTP port on the real server.

   ```
   ServerIronADX(config-rs-web1)# port http
   ```

   Syntax: port http

3. Define a real server (2) with an IP address.

   ```
   ServerIronADX(config-rs-web1)# server real web2 1.1.1.2
   ```

   Syntax: server real <*vip-name*> <*ip-address*>

4. Define a real HTTP port on the real server and exit.

   ```
   ServerIronADX(config-rs-web2)# port http
   ```

Syntax: port http

```
ServerIronADX(config-rs-web2)# exit
```

Syntax: exit

5. Define a virtual server with an IP address.

```
ServerIronADX(config)# server virtual-name-or-ip csw-vip 1.1.1.100
```

Syntax: server virtual-name-or-ip <*vip-name-or-ip-address*> <*ip-address*>

6. Define a virtual HTTP port on the virtual server.

```
ServerIronADX(config-vs-csw-vip)#port http
```

Syntax: port http

7. Bind HTTP ports on real servers web1 and web2 to the virtual port HTTP.

```
ServerIronADX(config-vs-csw-vip)# bind http web1 http web2 http
```

Syntax: bind http <*real-server*> **http** <*vip-name*>

Creating a CSW rule and policy with HTTP TRL

1. Define a CSW rule to match a pattern in the HTTP header that contains the client name.

```
ServerIronADX(config)# csw-rule rule1 header Authorization pattern Basic
```

Syntax: csw-rule <*rule-name*> **header** <*Authentication*> **pattern** <*Basic*>

2. Define a CSW policy.

```
ServerIronADX(config)# csw-policy policy1
```

Syntax: csw-policy <*policy-name*>

3. Specify an action to apply HTTP TRL policy when the CSW rule is matched.

```
ServerIronADX(config-csw-policy1)# match rule1 http-trl p1
```

Syntax: match <*rule-name*> **http-trl <http-trl-policy-name>**

Enabling Layer 7 SLB

To configure Layer 7 SLB, follow these steps.

1. Bind the policy to a virtual HTTP port on the virtual server.

```
ServerIronADX(config-vs-csw-vip)# port http csw-policy policy1
```

Syntax: port http csw-policy <*policy-name*>

2. Enable CSW on the virtual port.

```
ServerIronADX(config-vs-csw-vip)# port http csw
```

Syntax: port http csw

Displaying HTTP TRL

This section describes how to display HTTP TRL information.

Display all HTTP TRL policies

To show all running configurations for HTTP TRL policies, use the following command.

```
ServerIronADX# show run http-trl-policy all
```

Syntax: show run http-trl-policy all

Example

```
ServerIronADX# show run http-trl all
!Building configuration...
!Current configuration : 124813 bytes
!
http-trl-policy "my-http-trl-policy-104"
 tftp 50.50.50.105 "http-trl-policy-104.txt"
 client-name "root1" max-conn 1
 client-name "root1" exceed-action reset
 client-name "root10" max-conn 1
 client-name "root10" exceed-action reset
 client-name "root11" max-conn 1
 client-name "root11" exceed-action reset
 client-name "root12" max-conn 1
 client-name "root12" exceed-action reset
 client-name "root13" max-conn 1
 client-name "root13" exceed-action reset
 client-name "root14" max-conn 1
 client-name "root14" exceed-action reset
 client-name "root15" max-conn 1
 client-name "root15" exceed-action reset
 client-name "root16" max-conn 1
 client-name "root16" exceed-action reset
 client-name "root17" max-conn 1
 client-name "root17" exceed-action reset...
```

Display HTTP TRL policy from index

To show a running configuration for an HTTP TRL policy starting from an index, enter the following command.

```
ServerIronADX# show run http-trl-policy my-http-trl-policy-104 2
```

Syntax: show run http-trl-policy <*policy-name*> <*index*>

Example

```
ServerIronADX# show run http-trl my-http-trl-policy-104 2
!Building configuration...
!Current configuration : 4261 bytes
 client-name "root11" max-conn 1
 client-name "root11" exceed-action reset
 client-name "root12" max-conn 1
 client-name "root12" exceed-action reset
 client-name "root13" max-conn 1
 client-name "root13" exceed-action reset
 client-name "root14" max-conn 1
 client-name "root14" exceed-action reset
 client-name "root15" max-conn 1
 client-name "root15" exceed-action reset
 client-name "root16" max-conn 1
 client-name "root16" exceed-action reset
```

```
 client-name "root17" max-conn 1
 client-name "root17" exceed-action reset
 client-name "root18" max-conn 1
 client-name "root18" exceed-action reset
 client-name "root19" max-conn 1
 client-name "root19" exceed-action reset
 client-name "root2" max-conn 1
 client-name "root2" exceed-action reset
 client-name "root20" max-conn 1...
```

Display HTTP TRL policy client

To show a running configuration for an HTTP TRL policy client, enter the following command.

```
ServerIronADX# show run http-trl-policy my-http-trl-policy-104 root1
```

Syntax: show run http-trl-policy <*policy-name*> <*client-name*>

Example

```
ServerIronADX#show run http-trl my-http-trl-policy-104 root1
!Building configuration...
!Current configuration : 75 bytes
 client-name "root1" max-conn 1
 client-name "root1" exceed-action reset
```

Display HTTP TRL policy starting from index

To show a running configuration for an HTTP TRL policy starting from index for a specific number of entries, enter the following command.

```
ServerIronADX# show run http-trl-policy my-http-trl-policy-104 1 20
```

Syntax: show run http-trl-policy <*policy-name*> <*start-index*> <*number-of-entries*>

Example

```
ServerIronADX# show run http-trl my-http-trl-policy-104 1 20
!Building configuration...
!Current configuration : 1500 bytes
 client-name "root10" max-conn 1
 client-name "root10" exceed-action reset
 client-name "root11" max-conn 1
 client-name "root11" exceed-action reset
 client-name "root12" max-conn 1
 client-name "root12" exceed-action reset
 client-name "root13" max-conn 1
 client-name "root13" exceed-action reset
 client-name "root14" max-conn 1
 client-name "root14" exceed-action reset
 client-name "root15" max-conn 1
 client-name "root15" exceed-action reset
 client-name "root16" max-conn 1
 client-name "root16" exceed-action reset
 client-name "root17" max-conn 1
 client-name "root17" exceed-action reset
 client-name "root18" max-conn 1
```

```
 client-name "root18" exceed-action reset
 client-name "root19" max-conn 1
 client-name "root19" exceed-action reset
 client-name "root2" max-conn 1...
```

Display HTTP TRL policy matching a regular expression

To show a running configuration for an HTTP TRL policy matching a specific regular expression (regex), enter the following command.

NOTE
The syntax for regex is the same as for piping.

```
ServerIronADX# show run http-trl-policy my-http-trl-policy-109 regex ot1
```

Syntax: show run http-trl-policy <*policy-name*> **regex** < *regular expression*>

Example

```
ServerIronADX#show run http-trl my-http-trl-policy-104 regex ot1
!Building configuration...
!Current configuration : 825 bytes
 client-name "root1" max-conn 1
 client-name "root1" exceed-action reset
 client-name "root10" max-conn 1
 client-name "root10" exceed-action reset
 client-name "root11" max-conn 1
 client-name "root11" exceed-action reset
 client-name "root12" max-conn 1
 client-name "root12" exceed-action reset
 client-name "root13" max-conn 1
 client-name "root13" exceed-action reset
 client-name "root14" max-conn 1
 client-name "root14" exceed-action reset
 client-name "root15" max-conn 1
 client-name "root15" exceed-action reset
 client-name "root16" max-conn 1
 client-name "root16" exceed-action reset
 client-name "root17" max-conn 1
 client-name "root17" exceed-action reset
 client-name "root18" max-conn 1
 client-name "root18" exceed-action reset
 client-name "root19" max-conn 1...
```

Display HTTP TRL policy client index (MP)

To show an HTTP TRL policy client with a starting and ending index, enter the following command on the MP.

```
ServerIronADX# show http-trl policy my-http-trl-policy-103 0 10
```

Syntax: show http-trl policy <*policy-name*> <*start entry number*> <*end entry number*>

Example

```
ServerIronADX# show http-trl policy my-http-trl-policy-103 0 10
Policy Name:            my-http-trl-policy-103
                configured client count: 1
                total client count: 1
        Client name TDSWS/LoadRunner
                monitor-interval 1
                warning rate 10
                shutdown rate 20
                holddown interval 0
                exceed action: drop
                dynamic No
                max-conn track session 0
                trl track session 0
```

Syntax: show http-trl policy <*policy-name*> <*start entry number*> <*end entry number*>

NOTE
This command entered on the MP only displays configuration information and total entry count for this policy. The same command entered on the BP provides traffic status.

Display HTTP TRL policy client index (BP)

To show HTTP TRL policy client with a starting and ending index, use the following command on the BP.

```
ServerIronADX# show http-trl policy my-http-trl-policy-103 0 10
```

Syntax: show http-trl policy <*policy-name*> <*start entry number*> <*end entry number*>

Example

```
ServerIronADX# show http-trl policy my-http-trl-policy-103 0 100
Policy Name:            my-http-trl-policy-103
                configured client count: 1
                total client count: 2
        Client name V E'Væ\
                max-conn 50
                dynamic Yes
                max-conn track session 1
                trl track session 0
                HTTP_TRL_HIT    3278
                HTTP_TRL_PASS   1613
                HTTP_MAX_CONN_F 1665
                HTTP_TRL_DROP   1665
        Client name TDSWS/LoadRunner
                monitor-interval 1
                warning rate 10
                shutdown rate 20
                holddown interval 0
                exceed action: drop
                dynamic No
                max-conn track session 0
                trl track session 1
                HTTP_TRL_HIT    66352
                HTTP_TRL_PASS   39524
                HTTP_TRL_FAIL   26828
                HTTP_TRL_DROP   26828
ServerIronADX2/1#
ServerIronADX2/1# sh http-trl session 90.90.90.103  80 my-http-trl-policy-103
HTTP-MAX: V E'Væ\ config 50, current 50
HTTP-TRL: TDSWS/LoadRunner, config 2, attamp 3, hold 0, 1st 3089554, last 3092565
```

Display HTTP TRL policy for all client entries (BP)

To display HTTP TRL policy information for all client entries, enter the following command on the BP.

```
ServerIronADX2/1# show http-trl resource
```

Example

```
ServerIronADX2/1# show http-trl resource
Maximum client entry: 35000
Free client entry: 0
Total allocated client entry: 35000
Total freed client entry: 0
Maximum allocated client entry: 35000
Maximum client entry: 35000
Double free client entry: 0
Invalid free client entry: 0
Failed allocate client entry: 0
Double allocated client entry: 0
```

Downloading an HTTP TRL policy through TFTP

To download an HTTP TRL policy using TFTP, enter a command similar to the following.

```
ServerIronADX(config-http-trl-p1)# tftp 100.1.1.1 http-trl-config.txt
```

Syntax: **tftp** <*tftp-server-addr*> <*config-file-name*>

NOTE
You can save this command with write memory to automatically initiate a download for this policy after you reload. If you configure more than one policy for TFTP download, and a policy fails the download, the ServerIron does NOT retry, and the subsequent policy does not initiate a download. You must manually issue the command to do a TFTP download.

NOTE
When the total number of HTTP TRL entries exceeds 10k, the **show run time config** command cannot display an http trl-related configuration. You must use a text file to manage it.

NOTE
When any HTTP TRL policy client entry exceeds 1K, the **show run time config** command cannot display a detailed client entry for the HTTP TRL policy.

HTTP TRL policy commands

NOTE
You must configure client HTTP TRL before you configure the client exceed-limit

Client-name <client-name> monitor-interval

Use the **client-name** <*client-name*> **monitor-interval** option in the **http-trl-policy** configuration mode to set client rate limiting parameters.

Syntax: **[no] client-name** <*client-name*> **monitor-interval** <*interval-value*> <*warning-rate*> <*shutdown-rate*> <*holddown-interval*>

<*interval-value*>—specifies monitoring window in 100 ms unit.

<*warning-rate*>—specifies HTTP connection rate (per second) that causes a warning if exceeded.

<*shutdown-rate*>—specifies HTTP connection rate (per second) that causes a client to hold down.

<*holddown-interval*>—specifies the length of hold down period, if client exceeds rate limit in term of minutes.

NOTE
Value 0 means do not hold down. Hold down holds all traffic.

Example

```
ServerIronADX(config-http-trl-p1)# client-name c1 monitor-interval 1 10 20 0
```

Client-name <client-name> max-conn

Use the **client-name** <*client-name*> **max-conn** option in the **http-trl-policy** configuration mode to set client maximum connection parameters.

Syntax: **[no] client-name** <*client-name*> **max-conn** <*max-conn-value*>

<*max-conn-value*>—specifies maximum number of connections client can setup.

Example

```
ServerIronADX(config-http-trl-p1)# client-name c1 max-conn 10
```

NOTE
You must set the client HTTP max-conn configuration before you configure the client exceed-action.

NOTE
Max-conn currently supports only HTTP/1.0.

Client-name <client-name> exceed-action

Use the **client-name** <*client-name*> **exceed-action** option in the **http-trl-policy** configuration mode to set the action to take if a client exceeds the configured rate limit,.

Syntax: [no] client-name <*client-name*> **exceed-action [reset | drop]**

[reset | drop] specifies client request be reset or dropped if exceeds limit.

Example

```
ServerIronADX(config-http-trl-p1)# client-name c1 exceed-action [reset]
```

Syntax: [no] client-name <*client-name*> **exceed-action redirect** <*domain*> <*url*> **[port]**

<*domain*> and <*url*>—specifies client request to be redirected to this new URL, if limit is exceeded.

NOTE
Use an asterisk (*) to keep the same domain or url. This does not apply if the client is using HTTP 1.0.

```
ServerIronADX(config-http-trl-p1)# client-name c1 exceed-action redirect * /new
exceed.html http
```

NOTE
The same domain is used in the incoming packet.

The optional [port] specifies the new TCP port number for the redirected URL.

```
ServerIronADX(config-http-trl-p1)# client-name c1 exceed-action redirect
www.yahoo.com exceed.html http
```

Default monitor-interval

Use the **default monitor-interval** option in the **http-trl-policy** configuration mode to set default rate limiting parameters.

Syntax: [no] default monitor-interval <*interval-value*> <*warning-rate*> <*shutdown-rate*> <*holddown-interval*>

- <*interval-value*>—specifies monitoring window in 100 ms unit.
- <*warning-rate*>—specifies HTTP connection rate (per second) that causes a warning if exceeded.
- <*shutdown-rate*>—specifies HTTP connection rate (per second) that causes a client to hold down.

- <*holddown-interval*>—specifies the length of hold down period, if client exceeds rate limit in term of minutes.

NOTE
Value 0 means do not hold down. Hold down holds all traffic.

Example

```
ServerIronADX(config-http-trl-p1)# default monitor-interval 1 10 20 0
```

Default max-conn

Use the **default max-conn** option in the **http-trl-policy** configuration mode to set default maximum connection parameters.

Syntax: [no] default max-conn <*max-conn-value*>

<*max-conn-value*>—specifies maximum number of connections client can setup.

Example

```
ServerIronADX(config-http-trl-p1)# default max-conn 10
```

NOTE
Max-conn currently supports only HTTP/1.0.

Default exceed-action

Use the **default exceed-action** option in the **http-trl-policy** configuration mode to set the action to take if a default exceeds the configured rate limit.

Syntax: [no] default exceed-action [reset | drop]

[reset | drop] specifies default request be reset or dropped if the limit is exceeded.

Example

```
ServerIronADX(config-http-trl-p1)# default exceed-action [reset | drop]
```

Syntax: [no] default exceed-action redirect <*domain*> <*url*> **[port]**

<*domain*> and <*url*>—specifies client request to be redirected to this new URL, if limit is exceeded.

NOTE
Use an asterisk (*) to keep the same domain or url.

```
ServerIronADX(config-http-trl-p1)# default exceed-action redirect *
/new/exceed.html http
```

NOTE
The same domain is used in the incoming packet.

The optional [port] specifies the new TCP port number for the redirected URL.

```
ServerIronADX(config-http-trl-p1)# default exceed-action redirect  www.yahoo.com
/exceed.html http
```

Logging for DoS Attacks

The following sections describe how to enable logging of DoS attacks.

Configuration commands

Use the following commands to enable logging of TCP connection rate and attack rate.

Syntax: [no] ip tcp conn-rate <*rate*> **attack-rate** <*rate*>

Syntax: [no] ip tcp conn-rate-change <*percentage*> **attack-rate** <*percentage*>

Syntax: [no] server max-conn-trap <*seconds*>

Parameters

The **conn-rate** <*rate*> parameter specifies a threshold for the number of global TCP connections per second that are expected on the ServerIron. A global TCP connection is defined as any packet that requires session processing. For example, 1 SLB, 1 TCS, and 1 SYN-Guard connection would equal 3 global TCP connections, since there are three different connections that require session processing.

NOTE
The ServerIron ADX counts only the new connections that remain in effect at the end of the one second interval. If a connection is opened and terminated within the interval, the ServerIron ADX does not include the connection in the total for the server.

The **attack-rate** <*rate*> parameter specifies a threshold for the number of TCP SYN attack packets per second that are expected on the ServerIron.

Syslog entries are generated under the following circumstances:

- If the connection rate or attack rate on the ServerIron reaches 80% of the configured threshold.
- If the connection rate or attack rate is still between 80% and 100% of the configured threshold 6 minutes after the last message.
- If the connection rate or attack rate exceeds 100% of the configured threshold.
- If the connection rate or attack rate exceeds 100% of the configured threshold, and has gone up by the configured rate change percentage.
- One minute after the last message indicating that the connection rate or attack rate still exceeds 100% of the configured threshold, and has gone up by the configured rate change percentage.
- Three minutes after the last message, if the connection rate or attack rate is still between 80% and 100% of the configured threshold, and has gone up by the configured rate change percentage.

The **server max-conn-trap** <*seconds*> command specifies the number of seconds that elapse between traps, where <*seconds*> can be from 1 to 300. The default is 30.

Example

```
ServerIronADX(config)# ip tcp conn-rate 10000 attack-rate 10000
ServerIronADX(config)# ip tcp conn-rate-change 50 attack-rate 100
ServerIronADX(config)# server max-conn-trap 30
```

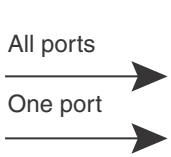
show server conn-rate

Use **show server conn-rate** to display the global TCP connection rate (per second) and TCP SYN attack rate (per second). This command reports global connection rate information for the ServerIron as well as for each real server.

```
ServerIronADX# show server conn-rate
Avail. Sessions      =     524286  Total Sessions       =     524288
Total C->S Conn      =          0  Total S->C Conn      =          0
Total Reassign       =          0  Unsuccessful Conn    =          0
last conn rate       =          0  max conn rate        =          0
last TCP attack rate =          0  max TCP attack rate  =          0
SYN def RST          =          0  SYN flood            =          0
Server State - 1:enabled, 2:failed, 3:test, 4:suspect, 5:grace_dn, 6:active

Real Server     State   CurrConn    TotConn   LastRate   CurrRate    MaxRate
rs1                 3          0          0          0          0          0
```

Maximum connections

Use max-conn to set the number of maximum connections on a global real server level (all ports) or a single port.




```
!
server real rs1 10.10.1.30
 max-conn 1200
 port http
 port http max-conn 1000
 port http url "HEAD /"
!
```

clear statistics dos-attack

Use clear statistics dos-attack to reset counters for ICMP and TCP SYN packet burst thresholds, as displayed by show statistics dos-attack.

Example

```
ServerIronADX# clear statistics dos-attack
ServerIronADX# show statistics dos-attack
```

NOTE

The above commands are used to reset and verify counters for ICMP and TCP SYN packet burst thresholds. The ServerIron ADX has introduced more a powerful feature to detect and block DoS attacks. Please refer to the chapter titled: "Syn-Proxy and DoS Protection" on page 113 to view details about verifying and clearing DOS-attack counters and filters.

Maximum concurrent connection limit per client

This feature restricts each client to a specified number of connections, based on the client's subnet, to prevent any one client from using all available connections.

Limiting the number of concurrent connections per client

This feature restricts each client to a specified number of concurrent connections, based on the client's subnet, to prevent any one client from using all available connections.

You associate a configured client subnet with a maximum permissible connection value. The association is stored in the ServerIron by means of a Dynamic Prefix (DP) trie. The key stored in the DP trie is the associated maximum connection value. The choice of the DP trie for storing the client subnet allows to define different prefix lengths and subnet masks for each client subnet. Since the DP trie lookup returns the longest prefix match, it is not required that all configured client subnets should have the same subnet mask.

Configuring the max connection limit per client consists of the following tasks:

- Configure the maximum connections allowed per client address or prefix
- Applying configured number of maximum connections to a specific VIP

Configure the maximum number of connections

1. Begin by creating a policy set or group by entering commands such as the following.

   ```
   ServerIronADX(config)#client-connection-limit max-conn1
   ```

   Syntax: [no] client-connection-limit <*name*>

   Enter a name for the policy set or group for <*name*>.

   Use the **no** form of the command to delete the policy group.

   After creating a name, the CLI changes to the config-client-max-conn level.

2. Next, create the policy for maximum number of connections using one of the following methods.

Create a policy for the maximum number of connections for specific clients

To set a maximum number of connections for a clients in a subnet, enter the a command such as the following.

```
ServerIronADX(config)# client-connection-limit max-conn1
ServerIronADX(config-client-max-conn)# max-conn 100.1.1.0 255.255.255.0 10
```

In the example above, clients with IP addresses in the 100.1.1.0 subnet will be allowed only 10 connections.

Syntax: [no] max-conn [<*client-ip-address*> <*client-subnet-mask*> <*max-connections*>

Enter the clients' IP address and subnet mask for <*client-ip-address*> <*client-subnet-mask*>

Enter a number from 0 to any value for <*max-connections*>. There is not default for this parameter.

Specifying a maximum number of connections for clients not specified in a policy

You can specify a default maximum number of connections for all clients that are not specified in any max connection group by entering a command such as the following.

```
ServerIronADX(config)# client-connection-limit max-conn1
ServerIronADX(config-client-max-conn)# max-conn default 10
```

In this example, all clients not specified in any max connection group will have a maximum of 10 connections.

Syntax: [no] max-conn [<*client-ip-address*> <*client-subnet-mask*> **default** <*max-connections*>

Enter a default maximum number of connections for <*max-connections*>

Excluding clients from maximum connection policy

If you want certain clients to be excluded from any maximum connection policies, enter a command such as the following.

```
ServerIronADX(config)# client-connection-limit max-conn1
ServerIronADX(config-client-trl)# max-conn 100.1.4.0 255.255.255.0 exclude
```

In this example, clients in the 100.1.4.0 subnet will be excluded for any maximum connection rules.

Syntax: [no] max-conn [<*client-ip-address*> <*client-subnet-mask*> **exclude**

Displaying the maximum number of connections for clients that are currently connected

To show the maximum number connection policy for a client that is currently connected, enter command such as the following on the barrel processor (BP) console.

```
ServerIronADX1# show conn pass1 0
Max Count: 2500 Total Count: 55
IP address Mask config hit denied
0.0.0.0 0.0.0.0 10 0 0
120.20.1.0 255.255.255.192 12 0 0
120.20.1.16 255.255.255.240 15 0 0
120.20.1.21 255.255.255.255 exclude 0 0
120.20.1.23 255.255.255.255 exclude 0 0
120.20.1.24 255.255.255.255 15 20 5
Current connections:
VIP 20.20.1.6: 15
120.20.1.25 255.255.255.255 exclude 0 0
120.20.1.27 255.255.255.255 exclude 20 0
Current connections:
VIP 20.20.1.6: 20
120.20.1.29 255.255.255.255 exclude 0 0
120.20.1.30 255.255.255.255 15 20 5
Current connections:
VIP 20.20.1.6: 15
120.20.1.33 255.255.255.255 exclude 20 0
ServerIronADX1#
```

Syntax: show connection-limit <*name*> <*offset*>

Enter the name of the max connection policy for <*name*>.

Enter the starting entry for <*offset*>

Binding the policy to a VIP

After creating a maximum connection policy, bind it to a VIP by entering commands such as the following.

```
ServerIronADX(config)#server virtual-name-or-ip virt-2
ServerIronADX(config-vs-virt-2)#client-max-conn-limit max-conn1
```

Syntax: [no] client-max-conn-limit <*name*>

Enter the name of the max connection policy for <*name*>.

NOTE
When the policy is bound to a VIP, the policy limits the number of connections that a client can have on any real server on the network.

Firewall load balancing enhancements

This section contains the following sections:

Enabling firewall strict forwarding

To enable load balancing only when traffic is going to a firewall, use the following command.

```
ServerIronADX(config)# server fw-strict-fwd
```

Syntax: server fw-strict-fwd

Use the **server fw-strict-fwd** command in the global configuration mode. Without this command, when the ServerIron receives traffic that matches the firewall flow session and the traffic is not received from a firewall, then the ServerIron assumes that it needs to be load balanced to a firewall.

This command checks to ensure that traffic is going to a firewall and only then does the ServerIron load balance it to a firewall.

Enabling firewall VRRPE priority

To configure VRRPE state to track the firewall group state, use the following command.

```
ServerIronADX(config)# server fw-g 2
ServerIronADX(config-tc-2)#fw-vrrpe-priority
ServerIronADX(config-tc-2)#
```

Syntax: fw-vrrpe-priority <*priority*>

Use the **fw-vrrpe-priority** command in the fw-group configuration mode. <*priority* > is the VRRPE priority associated with current firewall group state. Valid values are 1 to 255.

NOTE
This command can be used with the **track-fw-group** command below to force VRRPE state to track the firewall group state for a specific vrid.

Enabling track firewall group

To enable track-fw-group to track the firewall group state, use the following commands.

```
ServerIronADX(config)#int ve 1
ServerIronADX(config-vif-1)# ip vrrp-e vrid 1
ServerIronADX(config-vif-1-vrid-1)# track-fw-group
```

Syntax: track-fw-group <*group-num*>

Use the **track-fw-group** command under the VRRPE config level. <*group-num*> is the firewall group that needs to be tracked for this VRRPE. This command is used along with the fw-vrrpe-priority command to force VRRPE state to track the FW group state. This command works similar to the **track-port** command. When the firewall group state is STANDBY, then the VRRPE current priority is decremented by the fw-vrrpe-priority specified under that firewall group. It is recommended that you track only the firewall group state and no other port, because tracking firewall group state automatically tracks the router ports, firewall paths, and more.

Enabling firewall session sync delay

To enable server fw-sess-sync-delay, use the following command.

```
ServerIronADX(config)#server fw-sess-sync-delay 10
```

Syntax: server fw-sess-sync-delay <*secs*>

Use the **server fw-sess-sync-delay** command added at the global config level. <*secs*> is the number of seconds to delay the fast session sync after one of the ServerIrons is reloaded in HA FWLB. Valid values range from 1 to 100. This command can be useful in configurations where many real servers or firewalls are configured.

Syn-cookie threshhold trap

To configure the syn cookie attack rate threshold, use the following command.

```
ServerIronADX(config)# server syn-cookie-attack-rate-threshold 10000
```

Syntax: syn-cookie-attack-rate-threshold <*threshhold*>

The <*threshhold*> variable is a decimal number ranging from 1 to 10000000.

If the current syn cookie attack rate is larger than the syn cookie attack rate threshhold, the snTrapSynCookieAttackThreshReached trap is generated.

To configure the snTrapSynCookieAttackThreshReached trap interval, use the following command.

```
ServerIronADX(config)# server max-conn-trap-interval 10
```

Syntax: server max-conn-trap-interval <*decimal number in seconds*>

The <*seconds*> variable is a decimal number in seconds. The default value is 60 seconds.

Service port attack protection in hardware

A ServerIron can be enabled to deny traffic that is destined to VIP address but to a port that is not defined under a VIP. Such traffic can be dropped in hardware without impacting the MP or BP CPUs.

NOTE
VIP protection works for IPv4 VIPs alone and cannot be enabled for IPv6 VIPs.

You can enable this feature globally by entering the following command.

```
ServerIronADX(config)# server vip-protection
```

Syntax: [no] server vip-protection

Once enabled, the VIP protection applies to all existing and new VIP configurations.

If you want to enable the feature on individual VIPs, enter the following command.

```
ServerIronADX(config)# server virtual-name-or-ip v1
ServerIronADX(config-vs-v1)# vip-protection
```

NOTE
A reload is required for VIP protection to take effect when enabled on a global level using the **server vip-protection** command.

Syntax: [no] vip-protection

VIP protection adds CAM entries for each defined virtual port associated with each VIP. An additional CAM entry is defined for ICMP traffic destined to each VIP. An entry to drop the traffic is also added in the CAM for each VIP, which makes sure that traffic destined to any destination port other than the virtual ports is dropped by hardware.

NOTES:

- VIP protection does not support complex protocols such as FTP, TFTP, MMS, RTSP, SIP, that establish data connections based on the information exchanged on control channel.
- VIP protection cannot be enabled on a VIP that is part of a dynamic NAT address pool.
- VIP protection cannot be used along with features that require binding of virtual default port to real server default port.

Traffic segmentation

The traffic segmentation feature allows you to create segmentation among multiple L4-7 SLB domains of a single ServerIron ADX. The purpose of this feature is to ensure that traffic from one SLB domain to another SLB domain goes through the upstream firewall and does not get switched locally. This can be accomplished using either of the following methods:

- VLAN bridging
- Using the server use-session-for-vip-mac

These features help meet some of the security requirements for PCI compliance.

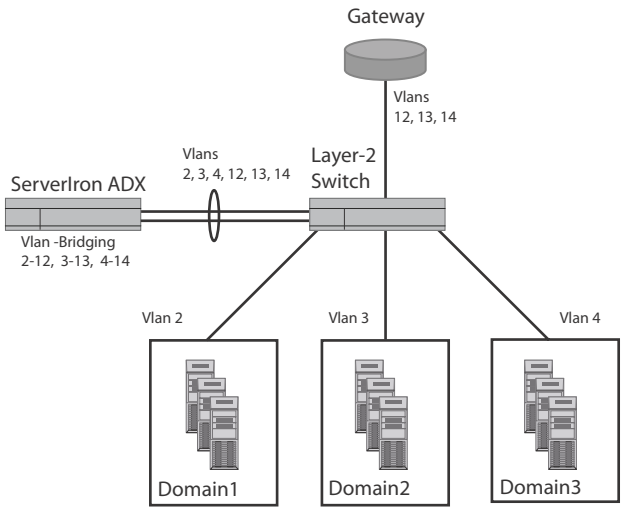
VLAN bridging

The VLAN bridging feature allows you to bridge together two VLANs so that packets will be layer-2 switched from one VLAN to the other. When two VLANs are bridged together, all packets received on one VLAN are translated to the other VLAN and switched.

When used for creating Layer-2 segmentation among SLB domains, this feature ensures that traffic from one SLB domain destined to another SLB domain goes through the upstream gateway and is not switched locally. This ensures that every packet between a client and server has to go through the ServerIron ADX for load-balancing.

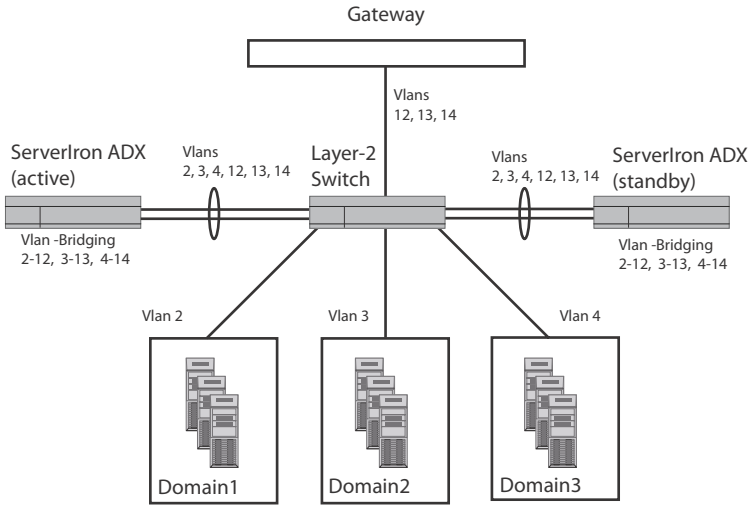
Figure 1 is an example of the VLAN bridging feature deployed in a one-armed topology. In this example when traffic from “Domain1” is bound for“Domain2” it is translated from VLAN 2 to VLAN 12 at the ServerIron ADX. It is then able to reach the “Gateway” on VLAN 12. The return traffic from the “Gateway” leaves on VLAN 13 and is translated to VLAN 3 at the ServerIron ADX. It is then able to reach “Domain2” on VLAN 3.

FIGURE 1 VLAN bridging in a one-armed topology



The topology described in Figure 1 can be implemented in the hot-standby configuration as shown in Figure 2.

FIGURE 2 VLAN bridging in a one-armed topology in High Availability configuration (hot-standby)



Considerations when configuring VLAN bridging

The following considerations apply when configuring VLAN bridging:

- Up to 64 unique-pair VLAN bridges can be configured.
- A VLAN cannot be part of two different VLAN bridges.
- Two VLANs forming a bridge must have the same set of member ports on the ServerIron ADX where they are joined.
- The Control VLAN (4094) and system default VLAN cannot be used for VLAN bridging.
- The hot-standby scenario is the only High Availability configuration supported with VLAN bridging. In a hot-standby scenario with one-armed topology, after fail over, the existing session may not be continued if the Layer-2 Switch in the middle cannot learn the MAC address of the Gateway through the newly-active ServerIron ADX in time.
- VLAN bridging is only supported with switch code. It is not supported with the ServerIron ADX router code.
- VLAN bridging is not supported with the SYN-proxy feature.
- All ports within a VLAN bridge must be tagged members of a VLAN and its associated bridged VLAN.
- MAC learning is shared for VLANs that are bridged together.

Configuring VLAN bridging

The **vlan-bridge** command is used to configure VLAN bridging. To configure VLAN 10 and VLAN 12 for VLAN bridging, use the following command.

```
ServerIron(config)# vlan-bridge 10 12
```

Syntax: [no] vlan-bridge <VLAN-number> <VLAN-number>

The <VLAN-number> variables specify the pair of VLANs that you want to create VLAN bridging for.

NOTE
Once a bridge is created between two VLANs, the VLAN configuration mode for those VLANs is disabled. You must remove a VLAN bridge if you want to make any changes to a VLAN contained within the VLAN bridge.

Example

The following example configures two VLANs with each containing the same ports and a VLAN bridge configured between them.

```
ServerIron(config)# vlan 222 by port
ServerIron(config-vlan-222)# tagged ethernet 1 ethernet 4
ServerIron(config-vlan-222)# exit
ServerIron(config)# vlan 333 by port
ServerIron(config-vlan-333)# tagged ethernet 1 ethernet 4
ServerIron(config-vlan-333)# exit
ServerIron(config)# vlan-bridge 222 333
```

Displaying VLAN bridge information

You can display information about VLAN bridging using the **show vlan** and **show vlan-bridge** commands.

Using the **show vlan** command, a VLAN bridge is displayed as shown in the following.

```
ServerIron# show vlan
Total PORT-VLAN entries: 3
Maximum PORT-VLAN entries: 64
PORT-VLAN 1, Name DEFAULT-VLAN, Priority level0, Spanning tree Off
 Untagged Ports: 2  3  5  6  7  8  9 10
   Tagged Ports: None
   Uplink Ports: None
 DualMode Ports: None
PORT-VLAN 222, Bridge VLAN 333, Name [None], Priority level0, Spanning tree Off
 Untagged Ports: None
   Tagged Ports: 1  4
   Uplink Ports: None
 DualMode Ports: None
PORT-VLAN 333, Bridge VLAN 222, Name [None], Priority level0, Spanning tree Off
 Untagged Ports: None
   Tagged Ports: 1  4
   Uplink Ports: None
 DualMode Ports: None
```

Syntax: show vlan [<*vlan-id*> | **ethernet** <port.>]

Using the <*vlan-id*> variable limits the display to the single VLAN whose ID is specified.

Using the **ethernet** <*port*> option limits the display to VLANs configured on the specified port.

The contents of the display are defined in the following table.

TABLE 2 Display from show vlan command

This field...	Displays...
PORT-VLAN	The VLAN ID of the PORT VLAN configured.
Bridge VLAN	The VLAN ID of the associated bridge VLAN.
Name	The name of the VLAN as configured. If no name is configured, "{None]" is displayed
Priority level	The QoS priority as configured. If no priority value is configured the value displayed will be "0".
Spanning tree	Displays the value of spanning tree protocol on this VLAN. Values can be "On" or "Off".
Untagged Ports	Displays the untagged port members of the VLAN.
Tagged Ports	Displays the tagged port members of the VLAN.
Uplink Ports	Displays the uplink port members of the VLAN.
DualMode Ports	Displays the port members of the VLAN that are in dual mode.

You can use the **show vlan-bridge** command to show all of the bridged VLANs as follows.

```
ServerIron# show vlan-bridge
IN-VLAN          Bridge VLAN
 222               333
 333               222
```

Syntax: show vlan-bridge

The contents of the display are defined in the following table.

TABLE 3 Display from show vlan-bridge command

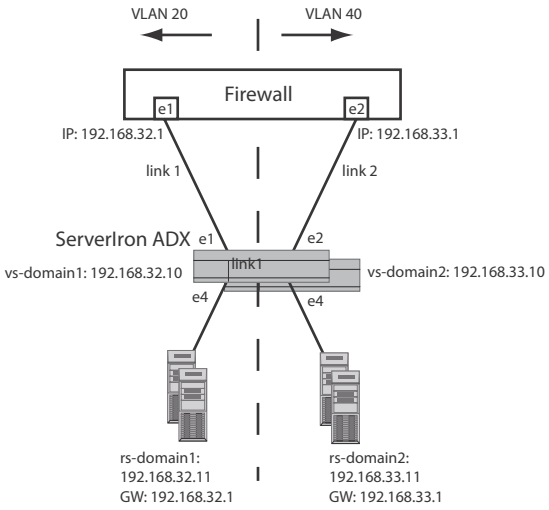
This field...	Displays...
IN-VLAN	The VLAN ID of the PORT VLAN configured.
Bridge VLAN	The VLAN ID of the associated bridge VLAN.

Traffic segmentation using the use-session-for-vip-mac command

By default, as long as there is a session match, packets with a destination IP address of a VIP are processed regardless of whether the destination MAC is addressed to the ServerIron ADX or not. With the **server use-session-for-vip-mac** command configured, only packets with a destination MAC address of the ServerIron ADX are processed. Packets with a destination IP address of a VIP but a destination MAC address not belonging to the ServerIron ADX are treated as pass-through traffic.

This feature is useful in traffic segmentation scenarios such as that shown in Figure 3. In the example, packets entering the ServerIron ADX from rs-domain1 bound for vs-domain2 would, by default, be switched at the ServerIron ADX to go directly to rs-domain2. If the **server use-session-for-vip-mac** command is configured on the ServerIron ADX, the packets are sent up to the firewall where they are subject to the security settings before being sent back down to the ServerIron ADX for forwarding to the VIP.

FIGURE 3 Traffic Segmentation



This feature is configured as shown in the following.

```
ServerIron(config)# server use-session-for-vip-mac
```

Syntax: [no] **server use-session-for-vip-mac**

DNS attack protection

The ServerIron ADX can be configured to provide DNS attack protection to VIP traffic. This protection is provided by performing a deep packet scan and then classifying DNS requests based on the following: query type, query name, RD flag or the DNSSEC “OK” bit in the EDNS0 header. Based on this classification, the following actions can be taken either individually or in combination: forward traffic to a specific server group, drop packets, log events or rate limit DNS traffic from the identified client.

Figure 4 displays a potential configuration of this feature. For this configuration, a DNS deep packet inspection with DNS filtering could be configured to perform the following actions.

Block specified types of DNS queries – for example:

- Block queries with the RD flag
- Block queries with the DNSSEC “OK” bit set.

Log specified types of DNS queries – for example:

- Log the number of queries to “www.mydomain.com”

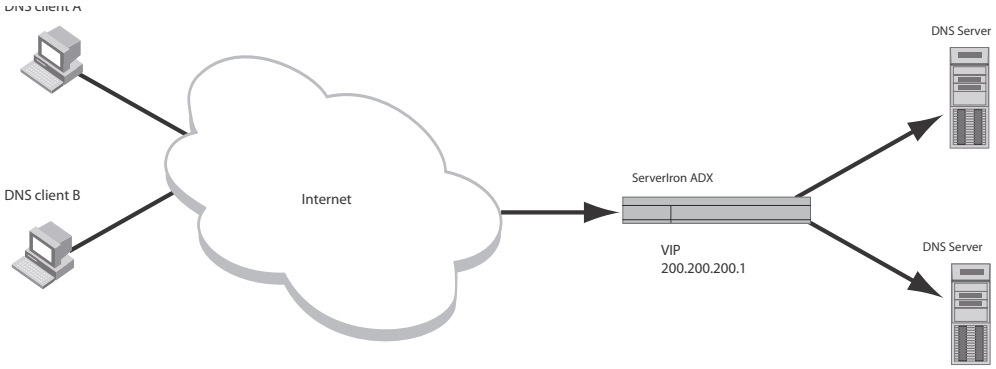
Redirect specified DNS queries to a different set of DNS servers – for example:

- Forward all requests with the DNSSEC “OK” bit to a separate set of servers.
- Forward all queries for the “ www.mydomain.com” to a different group of servers

Impose rate limiting for certain types of DNS queries per client.– for example:

- Rate limit queries to “ www.mydomain.com” for each client
- Rate limit the number of MX queries that a client can send.

FIGURE 4 DNS attack protection



Notes:

1. Only DNS requests using UDP transport (port 53) is supported.
2. If an incoming request matches an existing L4 session (including sticky sessions), DNS filtering will not apply on the request
3. Query not expected across multiple packet
4. When multiple queries are in a single DNS packet, only first RR will be processed
5. There is no csw dns rule to identify DNS Root requests.

Configuring DNS attack protection

Configuring DNS attack protection involves the following steps:

1. Create DNS DPI rules.

   In this step you specify the filtering parameters under a rule. A packet must match all of the filtering parameters defined under a rule to match the rule.

2. Create a DNS DPI policy and bind the rules to it.

   In this step you bind a rule to a policy and specify the action to be taken if a packet matches the rule.

3. Bind a DNS DPI policy to a Virtual port.

   In the final configuration step, you bind a policy to a virtual port. Then, all packets destined to that virtual are subject to the DNS DPI rules and policies defined in steps 1 and 2.

In addition, there are global commands that you can optionally configure to apply to all DNS attack protection configurations.

Defining DNS rules to filter packets

The DNS rules define the parameters that the DNS packets are filtered on. Rules can be defined for the following parameters:

- Query-name
- Query type
- RD flag
- DNS Sec bit

To define a rule, you must first define the rule and then define the DNS filtering rule parameters under it as shown.

```
ServerIron(config)# csw-rule rule1 udp-content dns
```

Syntax: [no] csw-rule <rule-name> udp-content dns

The <rule-name> variable specifies a name for the rule that must be unique across all CSW functionality. A maximum of 512 DNS DPI rules can be configured.

The filtering rule parameters are defined within the rule as shown. The rule parameters function as an inherent “AND” which means that all of the parameters must be met for the rule to be matched.

```
ServerIron(config)# csw-rule rule1 udp-content dns
ServerIron(config-csw-dns-rule-rule1) query-type MX
ServerIron(config-csw-dns-rule-rule1) query-name abc.com
ServerIron(config-csw-dns-rule-rule1) query-rd-flag on
ServerIron(config-csw-dns-rule-rule1) query-dnssec-ok off
```

Syntax: query-type <type>

The <type> variable specifies the DNS query type to match on.

Syntax: query-name <name>

The <name> variable specifies the name of the DNS query type to match on.

Syntax: query-rd-flag { on | off}

The **on** parameter is matched if the RD flag is set in the packet.

The **off** parameter is matched if the RD flag is not set in the packet.

Syntax: query-dnssec-ok { on | off}

The **on** parameter is matched if the DNSSEC bit is set in the packet.

The **off** parameter is matched if the DNSSEC bit is not set in the packet.

Order of Rule matching

Matching on the query-name is first attempted in the order of the length of the query-name. THis is followed by the rules without query-name (only if needed), in the order they were added to the policy. If two rules with query-name have the same length of the string, then the alphabetical order will take precedence. And, when two rules with query-name are exactly the same string, then the order in which the rules are added to the policy, will take precedence.

For example, initially the order of rules in a policy is:

1. Rule to match query-name www.brocade.com
2. Rule to match query-type A & query-RDflag ON

Adding a couple of new rules to match query-name www.mywebsite.com and to match query-type AAAA will rearrange the rules in policy as

1. Rule to match query-name www.brocade.com
2. Rule to match query-name www.mywebsite.com
3. Rule to match query-type A & query-RDflag ON
4. Rule to match query-type AAAA

The policy level configuration 'evaluate-generic-first' would reverse this default behavior by first matching the rules not based on query-names. In that case, same rules would be ordered as

1. Rule to match query-type A & query-RDflag ON
2. Rule to match query-type AAAA
3. Rule to match query-name www.brocade.com
4. Rule to match query-name www.mywebsite.com

Creating a DNS DPI policy and bind the rules to it

A DNS DPI policy specifies the action to take when a previously defined rule is matched. A DNS DPI policy is defined as shown.

```
ServerIron(config)# csw-policy DNSpolicy1 type dns-filter
```

Syntax: [no] csw-policy <policy-name> type dns-filter

The <policy-name> variable specifies a name for the CSW policy that must be unique across all CSW functionality.

NOTE

A maximum of 255 DNS policies can be configured on a ServerIron ADX. Also, the total number of rules that can be bound to a single policy is 512 and the global limit for binding rules to a policy is 2500. For example, if you bind 500 rules to each of 5 policies you will reach 2500 which is the global limit for binding rules to a policy.

Once a packet matches a configured filter, the following actions can be specified:

- drop
- Redirect to a server or server group
- rate-limit
- log (log is a secondary action and cannot be specified by itself)

The actions are configured within the DNS DPI policy as shown in the following.

```
ServerIron(config)# csw-policy DNSpolicy1 type dns-filter
ServerIron(config-csw-dns-policy-P1) match rule1 redirect 1 log
ServerIron(config-csw-dns-policy-P1) match rule2 drop log
ServerIron(config-csw-dns-policy-P1) match rule3 rate-limit monitor-interval 2
conn-rate 20 hold-down-time 2 log
ServerIron(config-csw-dns-policy-P1) default drop
```

Syntax: { match <rule-name> | default } {drop | redirect <group>| rate-limit monitor-interval <mon-value> conn-rate <conn-value> hold-down-time <hold-down-value> } { log | no-log }

If the **default** option is configured under a policy, DNS query packets that do not match any of the rules bound to that policy are acted on by the configured policy. In the example above, a DNS query that does not match rules rule1, rule2, and rule3 will be dropped.

The **drop** parameter directs the ServerIron ADX to drop any packets that match the filter.

The **redirect** parameter directs the ServerIron ADX redirect any packets that match the filter to a server or server group specified by <server-id> or <server-grp-id>

The **rate-limit** parameter directs the ServerIron ADX to rate limit packets that match the filter at the **monitor-interval** specified by the <mon-value> variable, the **conn-rate** specified by the <conn-value> and the **hold-down-time** specified by the <hold-down-value> variable.

The **log** parameter directs the ServerIron ADX to report the number of times that a rule has been matched within a 5 second interval. log is a secondary action and cannot be specified by itself.

Binding a DNS DPI policy to a Virtual port

To take effect, a DNS DPI policy must be bound to a virtual port. The following applies to this binding:

- a CSW DNS policy can only be applied to port DNS
- You can bind only one policy per virtual port
- You cannot bind a DNS policy to a virtual port if another CSW policy is already bound to port DNS.
- Once a DNS policy is bound to a port, any DNS query that comes to the virtual server will be matched against the rules bound to that policy and any associated action will be take on the match.

You can bind a DNS DPI policy to a virtual port as shown.

```
ServerIron(config) server virtual vip1 10.120.62.53
ServerIron(config-vs-vip1)# port dns csw-policy DNSpolicy1
ServerIron(config-vs-vip1)# port dns csw
```

Syntax: [no] port dns csw-policy <policy-name>

The <policy-name> variable specifies the name of the policy to be bound to a virtual port.

Syntax: [no] port dns csw

This command enables DNS content switching.

Configuring global commands for DNS attack protection

You can optionally configure the following to apply to all DNS attack protection configurations:

- Dropping all DNS packets that are fragmented
- Dropping all DNS packets with multiple queries
- Dropping all DNS packets that are malformed

To configure a ServerIron ADX to drop all DNS packets that are fragmented, use the **server dns-dpi drop-frag-pkts** command as shown.

```
ServerIron(config) server dns-dpi drop-frag-pkts
```

Syntax: [no] server dns-dpi drop-frag-pkts

To configure a ServerIron ADX to drop all DNS packets with multiple queries, use the **server dns-dpi drop-multiple-query-pkts** command as shown.

```
ServerIron(config) server dns-dpi drop-multiple-query-pkts
```

Syntax: [no] server dns-dpi drop-multiple-query-pkts

To configure a ServerIron ADX to drop all DNS packets that are malformed, use the **server dns-dpi drop-incomplete-malformed-pkts** command as shown.

```
ServerIron(config) server dns-dpi drop-incomplete-malformed-pkts
```

Syntax: [no] server dns-dpi drop-incomplete-malformed-pkts

Configuring the ADX to drop requests if servers in redirect actions are down

You can configure the ServerIron ADX to drop requests if servers in redirect actions are down as shown.

```
ServerIron(config-csw-pol-p1) dns-drop-on-fwd-fail
```

Syntax: [no] dns-drop-on-fwd-fail

Configuring the ADX to evaluate rules without query name first

You can configure the ServerIron ADX to evaluate rules without query name first as shown.

```
ServerIron(config-csw-pol-p1) evaluate-generic-first
```

Syntax: [no] evaluate-generic-first

Displaying DNS attack protection information

The following information can be displayed regarding DNS attack protection.

- DNS DPI policy counters
- IP addresses held down by a rate limit action

DIsplaying DNS DPI policy counters

DNS DPI policy counters can be displayed for a specified DNS policy as shown.

```
ServerIron# show csw-dns-policy p1
Rule Name       Action        Hit Count        Rate Limit Held Down
d2              redirect      0                0
d4              drop          0                0
d3              rate-limit    0                0
default         drop          0                0
```

You can display the DNS DPI policy counters for all DNS policies as shown.

```
ServerIron# show csw-dns-policy

Total Policies:3       Total Rules:6        Total Rule Actions:6
Policy Name   :p1                        Bind Count:2
Rule Name             Action          Hit Count         Rate Limit Held Down
d5                    redirect        0                 0
d1                    redirect        0                 0
d2                    redirect        0                 0
d3                    rate-limit      0                 0
default               drop            0                 0

Policy Name   :p2                       Bind Count:0
Rule Name     Action      Hit Count        Rate Limit Held Down

Policy Name   :p3                        Bind Count:0
Rule Name    Action       Hit Count         Rate Limit Held Down
d3           drop         0                 0
```

Syntax: show csw-dns-policy <policy-name>

The <policy-name> variable species a DNS policy that you want to display DNS DPI policy counters for.

CSW DNS DPI policy counters can be cleared for a specified DNS policy as shown.

```
ServerIron# clear csw-policy p1
```

Syntax: clear csw-policy <policy-name>

DIsplaying IP addresses held down by a rate limit action

IP addresses held down by a rate limit action can be displayed for an application processor (BP) from the rconsole as shown.

```
ServerIron ADX# rconsole 1 1
ServerIron ADX1/1# show security holddown
source         destination   vers   attempt   start   last   HD   time
30.30.30.4     0.0.0.3             3         45646   5646   N    1
```

1 DNS attack protection

Chapter

Access Control List 2

How ServerIron processes ACLs

This chapter describes the Access Control List (ACL) feature. ACLs allow you to filter traffic based on the information in the IP packet header. Depending on the Brocade device, the device may also support Layer 2 ACLs, which filter traffic based on Lay 2 MAC header fields.

You can use IP ACLs to provide input to other features such as distribution lists and rate limiting. When you use an ACL this way, use permit statements in the ACL to specify the traffic that you want to send to the other feature. If you use deny statements, the traffic specified by the deny statements is not supplied to the other feature.

There are two ways that IPv4 ACLs are processed in Brocade devices: in software and in hardware. This processing differs depending on the software release that you are running. These differences are described in the following sections.

Prior to release 12.3.01

Prior to release 12.3.01, IPv4 ACLs were processed as described in the following:

For deny actions:

All deny packets are dropped in hardware.

For permit actions:

For pass-through traffic, packets are processed in hardware.

For Layer 4 - 7 traffic, packets are forwarded to the BPs and the BPs perform the ACL processing.

Beginning with release 12.3.01 and later

Beginning with release 12.3.01, IPv4 ACLs are processed as described in the following:

For deny actions:

All deny packets are dropped in hardware.

For permit actions:

For pass-through traffic, packets are processed in hardware.

For Layer 4 - 7 traffic, packets are processed in hardware and then forwarded to the BPs. The BPs do not take any action on the ACLs.

Backwards compatibility option:

You can use the **ip flow-based-acl-enable** command to provide backwards compatibility for IPv4 ACL processing. If this command is configured, Layer 4 - 7 traffic, packets are processed in hardware and then forwarded to the BPs where the BPs also process the ACLs. This command is configured as shown in the following.

```
ServerIronADX(config)# ip flow-based-acl-enable
```

Syntax: ip flow-based-acl-enable

Rule-based ACLs

Some Brocade devices process the traffic that ACLs filter in hardware. This document refers to this type of ACLs as rule-based ACLs. These ACLs are programmed into hardware at startup or as a new ACL is entered.

Rule-based ACLs program the ACL entries you assign to an interface into Content Addressable Memory (CAM) space allocated for the port(s). Devices that use rule-based ACLs program the ACLs into the CAM entries and use these entries to permit or deny packets in the hardware, without sending the packets to the CPU for processing.

Rule-based ACLs are supported on physical interfaces, ve interfaces, trunk groups, and VIPs.

To configure a standard ACL and apply it to VIP, enter the following commands.

```
ServerIronADX(config)# access-list 1 deny host 209.157.22.26
ServerIronADX(config)# access-list 1 deny 209.157.29.12
ServerIronADX(config)# access-list 1 deny host IPHost1
ServerIronADX(config)# access-list 1 permit any
ServerIronADX 1000(config)#server virtual-name-or-ip vs-80
ServerIronADX 1000(config-vs-vs-80)#acl-id 1
ServerIronADX 1000(config-vs-vs-80)#write memory
```

Configuration guidelines for rule-based ACLs: general guidelines

Consider the following guidelines:

- Rule-based ACLs support only one ACL per port. The ACL of course can contain multiple entries (rules). For example, rule-based ACLs do not support ACLs 101 and 102 on port 1, but rule-based ACLs do support ACL 101 containing multiple entries.
- If you change the content of an ACL (add, change, or delete entries), you must remove and then reapply the ACL to all the ports that use it. Otherwise, the older version of the ACL remains in the CAM and continues to be used. You can easily re-apply ACLs using the **ip rebind-acl** <*num*> | <*name*> | **all** command. Refer to "Applying an ACLs to interfaces" on page 69.

NOTE
Brocade recommends that you also remove and reapply a changed ACL.

- ACL statistics are not supported with rule-based rate limiting.
- If you use the <*icmp-type*> parameter with an extended ACL, the device uses the CPU to filter packets using the ACL. The CPU is required to filter the ICMP message type.
- You can use PBR and rule-based ACLs on the same port. However, Brocade recommends that you use exactly the same ACL for each feature. Otherwise, it is possible for the ACL's Layer 4 CAM entry to be programmed incorrectly and give unexpected results.

How fragmented packets are processed

The descriptions for rule-based ACLs above apply to non-fragmented packets. The default processing of fragments by rule-based ACLs is as follows:

- The first fragment of a packet is permitted or denied using the ACLs. The first fragment is handled the same way as non-fragmented packets, since the first fragment contains the Layer 4 source and destination application port numbers. The device uses the Layer 4 CAM entry if one is programmed, or applies the interface's ACL entries to the packet and permits or denies the packet according to the first matching ACL.
- For other fragments of the same packet, one of the following occurs:
  - If the device has a CAM entry for the packet and has not been configured to send the fragments to the CPU, the device uses the CAM entry to forward the fragments in hardware.

    The fragments are forwarded even if the first fragment, which contains the Layer 4 information, was denied. Generally, denying the first fragment of a packet is sufficient, since a transaction cannot be completed without the entire packet. However, for stricter fragment control, you can send fragments to the CPU for filtering.
  - If the device is configured to send fragments to the CPU for filtering, the device compares the source and destination IP addresses to the ACL entries that contain Layer 4 information.
    - If the fragment's source and destination addresses exactly match an ACL entry that has Layer 4 information, the device assumes that the ACL entry is applicable to the fragment and permits or denies the fragment according to the ACL entry. The device does not compare the fragment to ACL entries that do not contain Layer 4 information.
    - If both the fragment's source and destination addresses do not exactly match an ACL entry, the device skips the ACL entry and compares the packet to the next ACL entry. This is true even if either the source or destination address (but not both) does exactly match an ACL entry.
    - If the source and destination addresses do not exactly match any ACL entry on the applicable interface, the device drops the fragment.

NOTE
By default, 10 Gigabit Ethernet modules also forward the first fragment instead of using the ACLs to permit or deny the fragment.

You can modify the handling of denied fragments. In addition, you can throttle the fragment rate on an interface that used rule-based ACLs. Refer to "Dropping all fragments that exactly match a flow-based ACL" on page 72 and "Enabling ACL filtering of fragmented packets" on page 73.

Default ACL action

The default action when no ACLs is configured on a device is to permit all traffic. However, once you configure an ACL and apply it to a port, the default action for that port is to deny all traffic that is not explicitly permitted on the port:

- If you want to tightly control access, configure ACLs consisting of permit entries for the access you want to permit. The ACLs implicitly deny all other access.

- If you want to secure access in environments with many users, you might want to configure ACLs that consist of explicit deny entries, then add an entry to permit all access to the end of each ACL. The software permits packets that are not denied by the deny entries.

Types of IP ACLs

Rule-based ACLs can be configured as standard or extended ACLs. A standard ACL permits or denies packets based on source IP address. An extended ACL permits or denies packets based on source and destination IP address and also based on IP protocol information.

Standard or extended ACLs can be numbered or named. Standard numbered ACLs have an idea of 1 – 99. Extended numbered ACLs are numbered 100 – 199. IDs for standard or extended ACLs can be a character string. In this document, ACLs with a string ID is called a named ACL.

ACL IDs and entries

ACLs consist of ACL IDs and ACL entries:

- **ACL ID** – An ACL ID is a number from 1 – 99 (for a standard ACL) or 100 – 199 (for an extended ACL) or a character string. The ACL ID identifies a collection of individual ACL entries. When you apply ACL entries to an interface, you do so by applying the ACL ID that contains the ACL entries to the interface, instead of applying the individual entries to the interface. This makes applying large groups of access filters (ACL entries) to interfaces simple.

  NOTE
  This is different from IP access policies. If you use IP access policies, you apply the individual policies to interfaces.

- **ACL entry** – An ACL entry are the filter commands associated with an ACL ID. These are also called “statements”. The maximum number of ACL entries you can configure is a system-wide parameter and depends on the device you are configuring. You can configure up to the maximum number of entries in any combination in different ACLs. The total number of entries in all ACLs cannot exceed the system maximum.
- Layer 3 switch code on devices can support up to 4096 ACL entries.

You configure ACLs on a global basis, then apply them to the incoming or outgoing traffic on specific ports. You can apply only one ACL to a port’s inbound traffic and only one ACL to a port’s outbound traffic. The software applies the entries within an ACL in the order they appear in the ACL’s configuration. As soon as a match is found, the software takes the action specified in the ACL entry (permit or deny the packet) and stops further comparison for that packet.

Support for up to 4096 ACL entries

You can configure up to 4096 ACL entries on devices that have enough space to hold a startup-config file that contains the ACLs.

For system-max configuration of 4096 ACL rules, the Ip access-group max-l4-cam parameter must be set to 4096. To configure the maximum ACL rule limit of 4096 ACL rules, the following must be set:

1. The system-max for Ip-filter-sys value must be set to 4096.

```
ServerIronADX(config)# system-max ip-filter-sys 4096
```

2. The Ip access-group max-l4-cam parameter must be set to 4096 on the interface that the ACL will be applied

```
ServerIronADX(config)# interface ethernet 1
ServerIronADX(config-if-e1000-1)# ip access-group max-l4-cam 4096
```

3. Execute the **write memory** command to save the running configuration to the startup-config reload the ServerIron ADX.

The actual number of ACLs you can configure and store in the startup-config file depends on the amount of memory available on the device for storing the startup-config. To store 4096 ACLs in the startup-config file requires at least 250K bytes, which is larger than the space available on a device's flash memory module.

You can load ACLs dynamically by saving them in an external configuration file on flash card or TFTP server, then loading them using one of the following commands.

```
copy tftp running-config <ip-addr> <filename>
ncopy tftp <ip-addr> <from-name> running-config
```

In this case, the ACLs are added to the existing configuration.

ACL entries and the Layer 4 CAM

Rule-based ACLs both use Layer 4 CAM entries.

Aging out of entries in the Layer 4 CAM

On a ServerIron ADX device, the device permanently programs rule-based ACLs into the CAM. The entries never age out.

Displaying the number of Layer 4 CAM entries

To display the number of Layer 4 CAM entries used by each ACL, enter the following command.

```
ServerIronADX(config)# show access-list all

Extended IP access list 100 (Total flows: N/A, Total packets: N/A, Total rule cam
use: 3)
permit udp host 192.168.2.169 any (Flows: N/A, Packets: N/A, Rule cam use: 1)
permit icmp any any (Flows: N/A, Packets: N/A, Rule cam use: 1)
deny ip any any (Flows: N/A, Packets: N/A, Rule cam use: 1)
```

Syntax: show access-list <*acl-num*> **|** <*acl-name*> **| all**

The Rule cam use field lists the number of CAM entries used by the ACL or entry. The number of CAM entries listed for the ACL itself is the total of the CAM entries used by the ACL's entries.

Specifying the maximum number of CAM entries for rule-based ACLs

For rule-based ACLs, you can adjust the allocation of Layer 4 CAM space for use by ACLs, on an IPC or IGC basis and on 10 Gigabit Ethernet modules. The new allocation applies to all the ports managed by the IPC or IGC or 10 Gigabit Ethernet module.

Most ACLs require one CAM entry for each ACL entry (rule). The exception is an ACL entry that matches on more than one TCP or UDP application port. In this case, the ACL entry requires a separate Layer 4 CAM entry for each application port on which the ACL entry matches.

Make sure you specify a maximum that is equal to or greater than the largest number of entries required by an ACL applied to any of the ports managed by the same IPC or IGC. For example, if port 1 will have an ACL that requires 250 entries, make sure 250 is the lowest number of entries you specify for any port on IPC 1 (the IPC that manages ports 1 – 24).

To specify the maximum number of CAM entries the device can allocate for rule-based ACLs, enter commands such as the following.

```
ServerIronADX(config)# interface ethernet 1/1
ServerIronADX(config-if-1/1)# ip access-group max-l4-cam 50
```

This command allows up to 50 ACL entries on each port managed by the IPC or IGC that manages port 1/1.

Syntax: [no] ip access-group max-l4-cam <*num*>

The <*num*> parameter specifies the number of CAM entries and can be from 10 – 2048. The default depends on the device.

The command is valid at the interface configuration level. However, the device applies the change to all ports managed by the same IPC or IGC. Regardless of the port number, when you save the change to the startup-config file, the CLI applies the command to the first port managed by the IPC or IGC. For example, if you enter the command on port 3, when you save the configuration change, the CLI enters the ip access-group max-l4-cam command under port 1 in the startup-config file.

NOTE
If you enter the command on more than one port managed by the same IPC or IGC, the CLI uses the value entered with the most-recent command for all the ports on the ICP or IGC.

Configuring numbered and named ACLs

When you configure ACLs, you can refer to the ACL by a numeric ID or by an alphanumeric name. The commands to configure numbered ACLs are different from the commands for named ACLs:

- If you refer to the ACL by a numeric ID, you can use 1 – 99 for a standard ACL or 100 – 199 for an extended ACL. This document refers to this ACL as *numbered ACL*.
- If you refer to the ACL by a name, you specify whether the ACL is a standard ACL or an extended ACL, then specify the name. This document refers to this ACL type as *named ACL*.

You can configure up to 100 standard numbered IP ACLs and 100 extended numbered IP ACLs. You also can configure up to 100 standard named ACLs and 100 extended named ACLs by number. Regardless of how many ACLs you have, the device can have a maximum of 4096 ACL entries, associated with the ACLs in any combination.

Configuring standard numbered ACLs

This section describes how to configure standard numbered ACLs with numeric IDs:

- For configuration information on named ACLs, refer to "Configuring standard or extended named ACLs" on page 62.
- For configuration information on extended ACLs, refer to "Configuring extended numbered ACLs" on page 56.

Standard ACLs permit or deny packets based on source IP address. You can configure up to 99 standard ACLs. There is no limit to the number of ACL entries an ACL can contain except for the system-wide limitation. For the number of ACL entries supported on a device, refer to "ACL IDs and entries" on page 52.

To configure a standard ACL and apply it to outgoing traffic on port 1/1, enter the following commands.

```
ServerIronADX(config)# access-list 1 deny host 209.157.22.26
ServerIronADX(config)# access-list 1 deny 209.157.29.12
ServerIronADX(config)# access-list 1 deny host IPHost1
ServerIronADX(config)# access-list 1 permit any
ServerIronADX(config)# int eth 1/1
ServerIronADX(config-if-1/1)# ip access-group 1 in
ServerIronADX(config)# write memory
```

The commands in this example configure an ACL to deny packets from three source IP addresses from being forwarded on port 1/1. The last ACL entry in this ACL permits all packets that are not explicitly denied by the first three ACL entries.

Standard ACL syntax

Syntax: [no] access-list <*num*> **deny | permit** <*source-ip*> **|** <*hostname*> <*wildcard*>

or

Syntax: [no] access-list <*num*> **deny | permit** <*source-ip*>/<*mask-bits*> **|** <*hostname*>

Syntax: [no] access-list <*num*> **deny | permit host** <*source-ip*> **|** <*hostname*>

Syntax: [no] access-list <*num*> **deny | permit any**

Syntax: [no] ip access-group <*num*> **in | out**

The <*num*> parameter is the access list number and can be from 1 – 99.

The **deny | permit** parameter indicates whether packets that match a policy in the access list are denied (dropped) or permitted (forwarded).

The <*source-ip*> parameter specifies the source IP address. Alternatively, you can specify the host name.

NOTE
To specify the host name instead of the IP address, the host name must be configured using the Brocade device's DNS resolver. To configure the DNS resolver name, use the **ip dns server-address...** command at the global CONFIG level of the CLI.

The <*wildcard*> parameter specifies the mask value to compare against the host address specified by the <*source-ip*> parameter. The <*wildcard*> is a four-part value in dotted-decimal notation (IP address format) consisting of ones and zeros. Zeros in the mask mean the packet's source address must match the <*source-ip*>. Ones mean any value matches. For example, the <*source-ip*> and <*wildcard*> values 209.157.22.26 0.0.0.255 mean that all hosts in the Class C sub-net 209.157.22.x match the policy.

If you prefer to specify the wildcard (mask value) in CIDR format, you can enter a forward slash after the IP address, then enter the number of significant bits in the mask. For example, you can enter the CIDR equivalent of "209.157.22.26 0.0.0.255" as "209.157.22.26/24". The CLI automatically converts the CIDR number into the appropriate ACL mask (where zeros instead of ones are the significant bits) and changes the non-significant portion of the IP address into ones. For example, if you specify 209.157.22.26/24 or 209.157.22.26 0.0.0.255, then save the changes to the startup-config file, the value appears as 209.157.22.0/24 (if you have enabled display of sub-net lengths) or 209.157.22.0 0.0.0.255 in the startup-config file.

If you enable the software to display IP sub-net masks in CIDR format, the mask is saved in the file in
"/<*mask-bits*>" format. To enable the software to display the CIDR masks, enter the **ip show-subnet-length** command at the global CONFIG level of the CLI. You can use the CIDR format to configure the ACL entry regardless of whether the software is configured to display the masks in CIDR format.

NOTE
If you use the CIDR format, the ACL entries appear in this format in the running-config and startup-config files, but are shown with sub-net mask in the display produced by the **show ip access-list** command.

The **host** <*source-ip*> | <*hostname*> parameter lets you specify a host IP address or name. When you use this parameter, you do not need to specify the mask. A mask of all zeros (0.0.0.0) is implied.

The **any** parameter configures the policy to match on all host addresses.

The in | out parameter specifies whether the ACL applies to incoming traffic or outgoing traffic on the interface to which you apply the ACL. You can apply the ACL to an Ethernet port. Note that the out option is not supported in the rule-based ACL mode.

Configuring extended numbered ACLs

This section describes how to configure extended numbered ACLs:

- For configuration information on named ACLs, refer to "Configuring numbered and named ACLs" on page 54.
- For configuration information on standard ACLs, refer to "Configuring standard numbered ACLs" on page 55.

Extended ACLs let you permit or deny packets based on the following information:

- IP protocol
- Source IP address or host name
- Destination IP address or host name
- Source TCP or UDP port (if the IP protocol is TCP or UDP)
- Destination TCP or UDP port (if the IP protocol is TCP or UDP)

The IP protocol can be one of the following well-known names or any IP protocol number from 0 – 255:

- Internet Control Message Protocol (ICMP)
- Internet Group Management Protocol (IGMP)
- Internet Gateway Routing Protocol (IGRP)
- Internet Protocol (IP)
- Open Shortest Path First (OSPF)
- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)

For TCP and UDP, you also can specify a comparison operator and port name or number. For example, you can configure a policy to block web access to a specific website by denying all TCP port 80 (HTTP) packets from a specified source IP address to the website's IP address.

To configure an extended access list that blocks all Telnet traffic received on port 1/1 from IP host 209.157.22.26, enter the following commands.

```
ServerIronADX(config)# access-list 101 deny tcp host 209.157.22.26 any eq telnet
ServerIronADX(config)# access-list 101 permit ip any any
ServerIronADX(config)# int eth 1/1
ServerIronADX(config-if-1/1)# ip access-group 101 in
ServerIronADX(config)# write memory
```

Here is another example of commands for configuring an extended ACL and applying it to an interface. These examples show many of the syntax choices.

```
ServerIronADX(config)# access-list 102 perm icmp 209.157.22.0/24
209.157.21.0/24
ServerIronADX(config)# access-list 102 deny igmp host rkwong 209.157.21.0/24
ServerIronADX(config)# access-list 102 deny igrp 209.157.21.0/24 host rkwong
ServerIronADX(config)# access-list 102 deny ip host 209.157.21.100 host
209.157.22.1
ServerIronADX(config)# access-list 102 deny ospf any any
ServerIronADX(config)# access-list 102 permit ip any any
```

The first entry permits ICMP traffic from hosts in the 209.157.22.x network to hosts in the 209.157.21.x network.

The second entry denies IGMP traffic from the host device named "rkwong" to the 209.157.21.x network.

The third entry denies IGRP traffic from the 209.157.21.x network to the host device named "rkwong".

The fourth entry denies all IP traffic from host 209.157.21.100 to host 209.157.22.1.

The fifth entry permits all packets that are not explicitly denied by the other entries. Without this entry, the ACL would deny all incoming or outgoing IP traffic on the ports to which you assign the ACL.

The following commands apply ACL 102 to the incoming traffic on port 1/2 and to the incoming traffic on port 4/3.

```
ServerIronADX(config)# int eth 1/2
ServerIronADX(config-if-1/2)# ip access-group 102 in
ServerIronADX(config-if-1/2)# exit
ServerIronADX(config)# int eth 4/3
ServerIronADX(config-if-4/3)# ip access-group 102 in
ServerIronADX(config)# write memory
```

Here is another example of an extended ACL.

```
ServerIronADX(config)# access-list 103 deny tcp 209.157.21.0/24 209.157.22.0/24
ServerIronADX(config)# access-list 103 deny tcp 209.157.21.0/24 eq ftp
209.157.22.0/24
ServerIronADX(config)# access-list 103 deny tcp 209.157.21.0/24 209.157.22.0/24 lt
telnet neq 5
ServerIronADX(config)# access-list 103 deny udp any range 5 6 209.157.22.0/24
range 7 8
ServerIronADX(config)# access-list 103 permit ip any any
```

The first entry in this ACL denies TCP traffic from the 209.157.21.x network to the 209.157.22.x network.

The second entry denies all FTP traffic from the 209.157.21.x network to the 209.157.22.x network.

The third entry denies TCP traffic from the 209.157.21.x network to the 209.157.22.x network, if the TCP port number of the traffic is less than the well-known TCP port number for Telnet (23), and if the TCP port is not equal to 5. Thus, TCP packets whose TCP port numbers are 5 or are greater than 23 are allowed.

The fourth entry denies UDP packets from any source to the 209.157.22.x network, if the UDP port number from the source network is 5 or 6 and the destination UDP port is 7 or 8.

The fifth entry permits all packets that are not explicitly denied by the other entries. Without this entry, the ACL would deny all incoming or outgoing IP traffic on the ports to which you assign the ACL.

The following commands apply ACL 103 to the incoming traffic on ports 2/1 and 2/2.

```
ServerIronADX(config)# int eth 2/1
ServerIronADX(config-if-2/1)# ip access-group 103 in
ServerIronADX(config-if-2/1)# exit
ServerIronADX(config)# int eth 2/2
ServerIronADX(config-if-2/2)# ip access-group 103 in
ServerIronADX(config)# write memory
```

Extended ACL syntax

Use the following syntax for configuring extended numbered ACLs.

Syntax: [no] access-list <*num*> deny | permit <*ip-protocol*> <*source-ip*> | <*hostname*> <*wildcard*> [<*operator*> <*source-tcp/udp-port*>] <*destination-ip*> | <*hostname*> [<*icmp-type*> | <*icmp-num*> | <*icmp-type-number*> <*icmp-code-number*>] <*wildcard*> [<*operator*> <*destination-tcp/udp-port*>] [established] [precedence <*name*> | <*num*>] [tos <*name*> | <*num*>] [ip-pkt-len <*value*>]

Syntax: [no] access-list <*num*> deny | permit host <*ip-protocol*> any any

Syntax: [no] ip access-group <*num*> in | out

The <*num*> parameter indicates the ACL number and be from 100 – 199 for an extended ACL.

The **deny | permit** parameter indicates whether packets that match the policy are dropped or forwarded.

The <*ip-protocol*> parameter indicates the type of IP packet you are filtering. You can specify a well-known name for any protocol whose number is less than 255. For other protocols, you must enter the number. Enter "?" instead of a protocol to list the well-known names recognized by the CLI.

The <*source-ip*> | <*hostname*> parameter specifies the source IP host for the policy. If you want the policy to match on all source addresses, enter **any**.

The <*wildcard*> parameter specifies the portion of the source IP host address to match against. The <*wildcard*> is a four-part value in dotted-decimal notation (IP address format) consisting of ones and zeros. Zeros in the mask mean the packet's source address must match the <*source-ip*>. Ones mean any value matches. For example, the <*source-ip*> and <*wildcard*> values 209.157.22.26 0.0.0.255 mean that all hosts in the Class C sub-net 209.157.22.x match the policy.

If you prefer to specify the wildcard (mask value) in Classless Interdomain Routing (CIDR) format, you can enter a forward slash after the IP address, then enter the number of significant bits in the mask. For example, you can enter the CIDR equivalent of "209.157.22.26 0.0.0.255" as "209.157.22.26/24". The CLI automatically converts the CIDR number into the appropriate ACL mask (where zeros instead of ones are the significant bits) and changes the non-significant portion of the IP address into zeros. For example, if you specify 209.157.22.26/24 or 209.157.22.26 0.0.0.255, then save the changes to the startup-config file, the value appears as 209.157.22.0/24 (if you have enabled display of sub-net lengths) or 209.157.22.0 0.0.0.255 in the startup-config file.

If you enable the software to display IP sub-net masks in CIDR format, the mask is saved in the file in "/<*mask-bits*>" format. To enable the software to display the CIDR masks, enter the **ip show-subnet-length** command at the global CONFIG level of the CLI. You can use the CIDR format to configure the ACL entry regardless of whether the software is configured to display the masks in CIDR format.

NOTE
If you use the CIDR format, the ACL entries appear in this format in the running-config and startup-config files, but are shown with sub-net mask in the display produced by the **show ip access-list** command.

The <*destination-ip*> | <*hostname*> parameter specifies the destination IP host for the policy. If you want the policy to match on all destination addresses, enter **any**.

The <*icmp-type*> | <*icmp-num*> parameter specifies the ICMP protocol type.

- This parameter applies only if you specified **icmp** as the <*ip-protocol*> value.
- If you use this parameter, the ACL entry is sent to the CPU for processing.
- If you do not specify a message type, the ACL applies to all types of ICMP messages.

The <*icmp-num*> parameter can be a value from 0 – 255.

The <*icmp-type*> parameter can have one of the following values, depending on the software version the device is running:

- any-icmp-type
- echo

- echo-reply
- information-request
- log
- mask-reply
- mask-request
- parameter-problem
- redirect
- source-quench
- time-exceeded
- timestamp-reply
- timestamp-request
- unreachable
- *<num>*

The *<operator>* parameter specifies a comparison operator for the TCP or UDP port number. This parameter applies only when you specify **tcp** or **udp** as the IP protocol. For example, if you are configuring an entry for HTTP, specify **tcp eq http**. You can enter one of the following operators:

- **eq** – The policy applies to the TCP or UDP port name or number you enter after eq.
- **gt** – The policy applies to TCP or UDP port numbers greater than the port number or the numeric equivalent of the port name you enter after gt.
- **lt** – The policy applies to TCP or UDP port numbers that are less than the port number or the numeric equivalent of the port name you enter after lt.
- **neq** – The policy applies to all TCP or UDP port numbers except the port number or port name you enter after neq.
- **range** – The policy applies to all TCP or UDP port numbers that are between the first TCP or UDP port name or number and the second one you enter following the range parameter. The range includes the port names or numbers you enter. For example, to apply the policy to all ports between and including 23 (Telnet) and 53 (DNS), enter the following: **range 23 53**. The first port number in the range must be lower than the last number in the range.
- **established** – This operator applies only to TCP packets. If you use this operator, the policy applies to TCP packets that have the ACK (Acknowledgment) or RST (Reset) bits set on (set to "1") in the Control Bits field of the TCP packet header. Thus, the policy applies only to established TCP sessions, not to new sessions. Refer to Section 3.1, "Header Format", in RFC 793 for information about this field.

  NOTE
  This operator applies only to destination TCP ports, not source TCP ports.

The *<tcp/udp-port>* parameter specifies the TCP or UDP port number or well-known name. You can specify a well-known name for any application port whose number is less than 1024. For other application ports, you must enter the number. Enter "?" instead of a port to list the well-known names recognized by the CLI.

The **in | out** parameter specifies whether the ACL applies to incoming traffic or outgoing traffic on the interface to which you apply the ACL. You can apply the ACL to an Ethernet port.

NOTE
The **out** option is not supported in the rule-based ACL mode.

The **precedence** <*name*> | <*num*> parameter of the **ip access-list** command specifies the IP precedence. The precedence option for of an IP packet is set in a three-bit field following the four-bit header-length field of the packet's header. You can specify one of the following:

- **critical** or **5** – The ACL matches packets that have the critical precedence. If you specify the option number instead of the name, specify number 5.
- **flash** or **3** – The ACL matches packets that have the flash precedence. If you specify the option number instead of the name, specify number 3.
- **flash-override** or **4** – The ACL matches packets that have the flash override precedence. If you specify the option number instead of the name, specify number 4.
- **immediate** or **2** – The ACL matches packets that have the immediate precedence. If you specify the option number instead of the name, specify number 2.
- **internet** or **6** – The ACL matches packets that have the internetwork control precedence. If you specify the option number instead of the name, specify number 6.
- **network** or **7** – The ACL matches packets that have the network control precedence. If you specify the option number instead of the name, specify number 7.
- **priority** or **1** – The ACL matches packets that have the priority precedence. If you specify the option number instead of the name, specify number 1.
- **routine** or **0** – The ACL matches packets that have the routine precedence. If you specify the option number instead of the name, specify number 0.

The **tos** <*name*> | <*num*> parameter of the **ip access-list** command specifies the IP ToS. You can specify one of the following:

- **max-reliability** or **2** – The ACL matches packets that have the maximum reliability ToS. The decimal value for this option is 2.
- **max-throughput** or **4** – The ACL matches packets that have the maximum throughput ToS. The decimal value for this option is 4.
- **min-delay** or **8** – The ACL matches packets that have the minimum delay ToS. The decimal value for this option is 8.
- **min-monetary-cost** or **1** – The ACL matches packets that have the minimum monetary cost ToS. The decimal value for this option is 1.

  NOTE
  This value is not supported on 10 Gigabit Ethernet modules.

- **normal** or **0** – The ACL matches packets that have the normal ToS. The decimal value for this option is 0.
- <*num*> – A number from 0 – 15 that is the sum of the numeric values of the options you want. The ToS field is a four-bit field following the Precedence field in the IP header. You can specify one or more of the following. To select more than one option, enter the decimal value that is equivalent to the sum of the numeric values of all the ToS options you want to select. For example, to select the **max-reliability** and **min-delay** options, enter number 10. To select all options, select 15.

The **ip-pkt-len** <*value*> parameter filters ICMP packets based on the IP packet length. The device uses the <*value*> to match the total length field in the IP header of ICMP packets. You can specify a value from 1 – 65535.

NOTE
This parameter applies only if you specified **icmp** as the <*ip-protocol*> value.

The **log** parameter enables SNMP traps and Syslog messages for packets denied by the ACL.

You can enable logging on ACLs and filters that support logging even when the ACLs and filters are already in use. To do so, re-enter the **ACL** or **filter** command and add the **log** parameter to the end of the ACL or filter. The software replaces the **ACL** or **filter** command with the new one. The new ACL or filter, with logging enabled, takes effect immediately.

Configuring standard or extended named ACLs

To configure a named IP ACL, use the following CLI method.

The commands for configuring named ACL entries are different from the commands for configuring numbered ACL entries. The command to configure a numbered ACL is **access-list**. The command for configuring a named ACL is **ip access-list**. In addition, when you configure a numbered ACL entry, you specify all the command parameters on the same command. When you configure a named ACL, you specify the ACL type (standard or extended) and the ACL number with one command, which places you in the configuration level for that ACL. Once you enter the configuration level for the ACL, the command syntax is the same as the syntax for numbered ACLs.

The following examples show how to configure a named standard ACL entry and a named extended ACL entry.

Configuration example for standard ACL

To configure a named standard ACL entry, enter commands such as the following.

```
ServerIronADX(config)# ip access-list standard Net1
ServerIronADX(config-std-nacl)# deny host 209.157.22.26 log
ServerIronADX(config-std-nacl)# deny 209.157.29.12 log
ServerIronADX(config-std-nacl)# deny host IPHost1 log
ServerIronADX(config-std-nacl)# permit any
ServerIronADX(config-std-nacl)# exit
ServerIronADX(config)# int eth 1/1
ServerIronADX(config-if-1/1)# ip access-group Net1 out
```

The commands in this example configure a standard ACL named “Net1”. The entries in this ACL deny packets from three source IP addresses from being forwarded on port 1/1. Since the implicit action for an ACL is “deny”, the last ACL entry in this ACL permits all packets that are not explicitly denied by the first three ACL entries. For an example of how to configure the same entries in a numbered ACL, refer to “Configuring standard numbered ACLs” on page 55.

Notice that the command prompt changes after you enter the ACL type and name. The “std” in the command prompt indicates that you are configuring entries for a standard ACL. For an extended ACL, this part of the command prompt is “ext“. The “nacl” indicates that are configuring a named ACL.

Syntax: ip access-list extended | standard <*string*> **|** <*num*>

The **extended** | **standard** parameter indicates the ACL type.

The <*string*> parameter is the ACL name. You can specify a string of up to 256 alphanumeric characters. You can use blanks in the ACL name if you enclose the name in quotation marks (for example, “ACL for Net1”). The <*num*> parameter allows you to specify an ACL number if you prefer. If you specify a number, you can specify from 1 – 99 for standard ACLs or 100 – 199 for extended ACLs.

NOTE
For convenience, the software allows you to configure numbered ACLs using the syntax for named ACLs. The software also still supports the older syntax for numbered ACLs. Although the software allows both methods for configuring numbered ACLs, numbered ACLs are always formatted in the startup-config and running-config files in using the older syntax, as follows.

```
access-list 1 deny host 209.157.22.26
access-list 1 deny 209.157.22.0 0.0.0.255
access-list 1 permit any
access-list 101 deny tcp any any eq http
```

The options at the ACL configuration level and the syntax for the **ip access-group** command are the same for numbered and named ACLs and are described in “Configuring standard numbered ACLs” on page 55.

Configuration example for extended ACL

To configure a named extended ACL entry, enter commands such as the following.

```
ServerIronADX(config)# ip access-list extended “block Telnet”
ServerIronADX(config-ext-nacl)# deny tcp host 209.157.22.26 any eq telnet
ServerIronADX(config-ext-nacl)# permit ip any any
ServerIronADX(config-ext-nacl)# exit
ServerIronADX(config)# int eth 1/1
ServerIronADX(config-if-1/1)# ip access-group “block Telnet” in
```

The options at the ACL configuration level and the syntax for the **ip access-group** command are the same for numbered and named ACLs and are described in “Configuring extended numbered ACLs” on page 56.

Displaying ACL definitions

To display the ACLs configured on a device, use the **show ip access-lists** command. Here is an example.

```
ServerIronADX(config)# show ip access-lists
Extended IP access list 101
    deny tcp host 209.157.22.26 host 209.157.22.26 eq http
```

Syntax: show ip access-lists [<*num*>]

The **show access-list** and **show ip access-list** commands have been updated to display ACL entries with line numbers.

Numbered ACL

For a numbered ACL, you can enter a command such as the following.

```
ServerIronADX(config)# show access-list 99 3
Standard IP access-list 99
deny 10.10.10.1
deny 192.168.1.13
permit any
```

Syntax: show access-list <*acl-number*> [<*line-number*>]

Enter the ACL' number for the <*acl-number*> parameter.

Determine from which line you want the displayed information to begin and enter that number for the <*line-number*> parameter.

Named ACL

For a named ACL, enter a command such as the following.

```
ServerIronADX(config)# ip show access-list standard melon 3
Standard IP access-list melon
deny host 5.6.7.8
deny 192.168.12.3
permit any
```

Syntax: show ip access-list <*acl-name*> | <*acl-number*> [<*line-number*>]

Enter the ACL name for the <*acl-name*> parameter or the ACL's number for <*acl-number*>.

Determine from which line you want the displayed information to begin and enter that number for the <*line-number*> parameter.

Displaying ACLs using keywords

You limit the displayed ACL entries to those that contain a specified keyword.

Numbered ACL

You may have the following numbered ACL.

```
ServerIronADX(config)# show access-list 99
Standard IP access-list 99
deny host 1.2.3.4
permit host 5.6.7.8
permit host 5.10.11.12
permit any
```

If you want to display ACL entries beginning with the entry that contains the keyword "5" enter the following command.

```
ServerIronADX(config)# show access-list 99 | begin 5
Standard IP access-list 99
permit host 5.6.7.8
permit host 5.10.11.12
permit any
```

Since the second entry is the first entry that contains the keyword "5", the display begins with line 2.

If you want to display only the ACL entries that contain the keyword "5" enter the following command.

```
ServerIronADX(config)#show access-list 99 | include 5
Standard IP access-list 99
permit host 5.6.7.8
permit host 5.10.11.12
```

The second and third entries in the ACL contain the keyword “5” and are displayed in the **show access-list**.

If you want to exclude ACL entries that contain a keyword from the show access-list display, enter the following command.

```
ServerIronADX(config)# show access-list 99 | exclude 5
Standard IP access-list 99
deny host 1.2.3.4
permit any
```

The second and third ACL entries are left out from the display.

Syntax: show access-list <*acl-number*> **| begin | exclude | include** <*keyword*>

Enter the ACL number for the <*acl-number*> parameter.

Use the **|** operator to indicate a keyword.

Enter the **begin** <*keyword*> parameter to start the display beginning with the first line containing the text that matches the keyword. For example, if you enter `begin Total`, the displayed information begins with the line containing the word “Total”.

Enter the **exclude** <*keyword*> parameter to exclude any lines containing text that match the keyword. For example, if you enter `exclude Total`, any line containing the word “Total” is excluded from the display.

Enter the **include** <*keyword*> display only those lines containing text that match the keyword. For example, if you enter `include Total`, any line containing the word “Total” is included in the display.

Named ACLs

You may have the following numbered ACL.

```
ServerIronADX(config)# show access-list melon
Standard IP access-list melon
deny host 1.2.3.4
permit host 5.6.7.8
permit host 5.10.11.12
permit any
```

If you want to display ACL entries beginning with the entry that contains the keyword “5” enter the following command.

```
ServerIronADX(config)# show access-list melon | begin 5
Standard IP access-list melon
permit host 5.6.7.8
permit host 5.10.11.12
permit any
```

Since the second entry is the first entry that contains the keyword “5”, the display begins with line 2.

If you want to display only the ACL entries that contain the keyword “5” enter the following command.

```
ServerIronADX(config)# show access-list melon | include 5
Standard IP access-list melon
permit host 5.6.7.8
permit host 5.10.11.12
```

The second and third entries in the ACL contain the keyword “5” and are displayed in the **show access-list**.

If you want to exclude ACL entries that contain a keyword from the show access-list display, enter the following command.

```
ServerIronADX(config)# show access-list melon | exclude 5
Standard IP access-list melon
deny host 1.2.3.4
permit any
```

The second and third ACL entries are left out from the display.

Syntax: show ip access-list <*acl-number*> **| begin | exclude | include** <*keyword*>

Enter the ACL’s number for the <*acl-number*> parameter.

Use the **|** operator to indicate a keyword.

Enter the **begin** <*keyword*> parameter to start the display beginning with the first line containing text that matches the keyword. For example, if you enter begin Total, the displayed information begins with the line containing the word “Total”.

Enter the **exclude** <*keyword*> parameter to exclude any lines containing text that match the keyword. For example, if you enter exclude Total, any line containing the word “Total” is excluded from the display.

Enter the **include** <*keyword*> display only those lines containing text that match the keyword. For example, if you enter include Total, any line containing the word “Total” is included in the display.

```
If ACL entries, for both numbered and named ACLs, have remarks, the display
will also include the remarks if they contain characters that match the
specified keywords. For example, ACL 99 might contain the following entries:

ServerIronADX(config)# show access-list 99
Standard IP access-list 99
   ACL Remark: Deny Building A
deny host 1.2.3.4
Permit First Floor Users
permit host 5.6.7.8
deny host 5.10.11.12
permit any

To show all entries containing the keyword “deny” you obtain the following
output:

ServerIronADX(config)#show access-list 99 | include deny
Standard IP access-list 99
   ACL Remark: Deny Building A
deny host 1.2.3.4
deny host 5.10.11.12
```

NOTE
All lines with the keyword “deny”, including remarks are included in the display.

Modifying ACLs

When you use the Brocade device's CLI to configure any ACL, the software places the ACL entries in the ACL in the order you enter them. For example, if you enter the following entries in the order shown below, the software always applies the entries to traffic in the same order.

```
ServerIronADX(config)# access-list 1 deny 209.157.22.0/24
ServerIronADX(config)# access-list 1 permit 209.157.22.26
```

If a packet matches the first ACL entry in this ACL and is therefore denied, the software does not compare the packet to the remaining ACL entries. In this example, packets from host 209.157.22.26 will always be dropped, even though packets from this host match the second entry.

You can use the CLI to reorder entries within an ACL by individually removing the ACL entries and then re-adding them. To use this method, enter "no" followed by the command for an ACL entry, and repeat this for each ACL entry in the ACL you want to edit. After removing all the ACL entries from the ACL, re-add them.

This method works well for small ACLs such as the example above, but can be impractical for ACLs containing many entries. Therefore, Brocade devices provide an alternative method. The alternative method lets you upload an ACL list from a TFTP server and replace the ACLs in the device's running-config file with the uploaded list. Thus, to change an ACL, you can edit the ACL on the file server, then upload the edited ACL to the device. You then can save the changed ACL to the device's startup-config file.

ACL lists contain only the ACL entries themselves, not the assignments of ACLs to interfaces. You must assign the ACLs on the device itself.

NOTE
The only valid commands that are valid in the ACL list are the **access-list** and **end** commands. The Brocade device ignores other commands in the file.

To modify an ACL by configuring an ACL list on a file server.

1. Use a text editor to create a new text file. When you name the file, use 8.3 format (up to eight characters in the name and up to three characters in the extension).

   NOTE
   Make sure the Brocade device has network access to the TFTP server.

2. Optionally, clear the ACL entries from the ACLs you are changing by placing commands such as the following at the top of the file.

   ```
   no access-list 1
   no access-list 101
   ```

   When you load the ACL list into the device, the software adds the ACL entries in the file after any entries that already exist in the same ACLs. Thus, if you intend to entirely replace an ACL, you must use the **no access-list** <*num*> command to clear the entries from the ACL before the new ones are added.

3. Place the commands to create the ACL entries into the file. The order of the separate ACLs does not matter, but the order of the entries within each ACL is important. The software applies the entries in an ACL in the order they are listed within the ACL. Here is an example of some ACL entries.

```
access-list 1 deny host 209.157.22.26 log
access-list 1 deny 209.157.22.0 0.0.0.255 log
access-list 1 permit any
access-list 101 deny tcp any any eq http log
```

The software will apply the entries in ACL 1 in the order shown and stop at the first match. Thus, if a packet is denied by one of the first three entries, the packet will not be permitted by the fourth entry, even if the packet matches the comparison values in this entry.

4. Enter the command **end** on a separate line at the end of the file. This command indicates to the software that the entire ACL list has been read from the file.
5. Save the text file.
6. On the Brocade device, enter the following command at the Privileged EXEC level of the CLI.

```
copy tftp running-config <tftp-ip-addr> <filename>
```

NOTE
This command will be unsuccessful if you place any commands other than access-list and end (at the end only) in the file. These are the only commands that are valid in a file you load using the **copy tftp running-config...** command.

7. To save the changes to the device's startup-config file, enter the following command at the Privileged EXEC level of the CLI.

```
write memory
```

Here is a complete example of an ACL configuration file.

```
no access-list 1
no access-list 101
access-list 1 deny host 209.157.22.26 log
access-list 1 deny 209.157.22.0 0.0.0.255 log
access-list 1 permit any
access-list 101 deny tcp any any eq http log
end
```

NOTE
Do not place other commands in the file. The Brocade device reads only the ACL information in the file and ignores other commands, including **ip access-group** commands. To assign ACLs to interfaces, use the CLI.

Displaying a list of ACL entries

The **show access-list** and **show ip access-list** commands displays ACL entries with line numbers.

Numbered ACLs

To display the contents of numbered ACLs, enter a command such as the following.

```
ServerIronADX# show access-list 99
Standard IP access list 99
deny host 1.2.4.5
deny host 5.6.7.8
permit any
```

Syntax: show access-list <*acl-num*> **| all**

Named ACLs

To display the contents of named ACLs, enter a command such as the following.

```
ServerIronADX# show ip access-list melon
Standard IP access list melon
deny host 1.2.4.5
deny host 5.6.7.8
permit any
```

Syntax: show ip access-list <*acl-num*> | <*acl-name*>

Applying an ACLs to interfaces

Configuration examples in the section "Configuring numbered and named ACLs" on page 54 show that you apply ACLs to interfaces using the **ip access-group** command. This section present additional information about applying ACLs to interfaces.

Reapplying modified ACLs

If you make an ACL configuration change, you must reapply the ACLs to their interfaces to place the change into effect.

An ACL configuration change includes any of the following:

- Adding, changing, or removing an ACL or an entry in an ACL
- Changing a PBR policy

To reapply ACLs following an ACL configuration change, enter the following command at the global CONFIG level of the CLI.

```
ServerIronADX(config)# ip rebind-acl all
```

Syntax: [no] ip rebind-acl <*num*> | <*name*> | **all**

ACL logging

You may want the software to log entries for ACLs in the syslog. This section present the how logging is processed by rule-based ACLs.

Rule-based ACLs do not support the **log** option. Even when rule-based ACLs are enabled, if an ACL entry has the **log** option, traffic that matches that ACL is sent to the CPU for processing. Depending on how many entries have the log option and how often packets match those entries, ACL performance can be affected.

If your configuration already contains ACLs that you want to use with rule-based ACLs, but some of the ACLs contain the **log** option, the software changes the ACL mode to flow-based for the traffic flows that match the ACL. Changing the mode to flow-based enables the device to send the matching flows to the CPU for processing. This is required because the CPU is needed to generate the Syslog message.

You can globally disable ACL logging without the need to remove the **log** option from each ACL entry. When you globally disable ACL logging, the ACL entries remain unchanged but the **log** option is ignored and the ACL can use the rule-based ACL mode. This enables you to use the ACLs in the rule-based ACL mode. You also can configure the device to copy traffic that is denied by a rule-based ACL to an interface. This option allows you to monitor the denied traffic without sending the traffic to the CPU.

To globally disable ACL logging, enter the following command at the global CONFIG level of the CLI.

```
ServerIronADX(config)# ip access-list disable-log-to-cpu
```

Syntax: [no] ip access-list disable-log-to-cpu

To re-enable ACL logging, enter the following command.

```
ServerIronADX(config)# no ip access-list disable-log-to-cpu
```

Syslog message for changed ACL mode

If the device changes the ACL mode from rule-based to flow-based, the device generates one of the following Syslog notification messages:

- ACL insufficient L4 session resource, using flow based ACL instead.
- ACL exceed max DMA L4 cam resource, using flow based ACL instead. Refer to "Specifying the maximum number of CAM entries for rule-based ACLs" on page 54.
- ACL insufficient L4 cam resource, using flow based ACL instead.

Copying denied traffic to a mirror port for monitoring

Although rule-based ACLs do not support ACL logging, you nonetheless can monitor the traffic denied by rule-based ACLs. To do so, attach a protocol analyzer to a port and enable the device to redirect traffic denied by ACLs to that port.

To redirect traffic denied by ACLs, enter the following command at the interface configuration level.

```
ServerIronADX(config-if-1/1)# ip access-group redirect-deny-to-interf
```

Syntax: [no] ip access-group redirect-deny-to-interf

Enter the command on the port to which you want the denied traffic to be copied.

NOTE
The software requires that an ACL has already been applied to the interface.

When you enable redirection, the deny action of the ACL entry is still honored. Traffic that matches the ACL is not forwarded.

Displaying ACL log entries

The first time an entry in an ACL permits or denies a packet and logging is enabled for that entry, the software generates a Syslog message and an SNMP trap. Messages for packets permitted or denied by ACLs are at the warning level of the Syslog.

When the first Syslog entry for a packet permitted or denied by an ACL is generated, the software starts an ACL timer. After this, the software sends Syslog messages every one to ten minutes, depending on the value of the timer interval. If an ACL entry does not permit or deny any packets during the timer interval, the software does not generate a Syslog entry for that ACL entry.

NOTE
For an ACL entry to be eligible to generate a Syslog entry for permitted or denied packets, logging must be enabled for the entry. The Syslog contains entries only for the ACL entries that deny packets and have logging enabled.

To display Syslog entries, enter the following command from any CLI prompt.

```
ServerIronADX(config)# show log
Syslog logging: enabled (0 messages dropped, 0 flushes, 0 overruns)
    Buffer logging: level ACDMEINW, 38 messages logged
    level code: A=alert C=critical D=debugging M=emergency E=error
                I=informational N=notification W=warning

Log Buffer (50 entries):

21d07h02m40s:warning:list 101 denied tcp 209.157.22.191(0)(Ethernet 4/18
0010.5a1f.77ed) -> 198.99.4.69(http), 1 event(s)
00d07h03m30s:warning:list 101 denied tcp 209.157.22.26(0)(Ethernet 4/18
0010.5a1f.77ed) -> 198.99.4.69(http), 1 event(s)
00d06h58m30s:warning:list 101 denied tcp 209.157.22.198(0)(Ethernet 4/18
0010.5a1f.77ed) -> 198.99.4.69(http), 1 event(s)
```

In this example, the two-line message at the bottom is the first entry, which the software immediately generates the first time an ACL entry permits or denies a packet. In this case, an entry in ACL 101 denied a packet. The packet was a TCP packet from host 209.157.22.198 and was destined for TCP port 80 (HTTP) on host 198.99.4.69.

When the software places the first entry in the log, the software also starts the five-minute timer for subsequent log entries. Thus, five minutes after the first log entry, the software generates another log entry and SNMP trap for denied packets.

In this example, the software generates the second log entry five minutes later.

The time stamp for the third entry is much later than the time stamps for the first two entries. In this case, no ACLs denied packets for a very long time. In fact, since no ACLs denied packets during the five-minute interval following the second entry, the software stopped the ACL log timer. The software generated the third entry as soon as the ACL denied a packet. The software restarted the five-minute ACL log timer at the same time. As long as at least one ACL entry permits or denies a packet, the timer continues to generate new log entries and SNMP traps every five minutes.

You can also configure the maximum number of ACL-related log entries that can be added to the system log over a one-minute period. For example, to limit the device to 100 ACL-related syslog entries per minute.

```
ServerIronADX(config)# max-acl-log-num 100
```

Syntax: [no] max-acl-log-num <*num*>

You can specify a number between 0 – 4096. The default is 256. Specifying 0 disables all ACL logging.

Displaying ACL statistics for flow-based ACLs

To display ACL statistics for flow-based ACLs, enter the following command.

```
ServerIronADX(config)# show ip acl-traffic

ICMP inbound packets received 400
ICMP inbound packets permitted 200
ICMP inbound packets denied 200
```

Syntax: show ip acl-traffic

The command lists a separate set of statistics for each of the following IP protocols:

- ICMP
- IGMP
- IGRP
- IP
- OSPF
- TCP
- UDP
- Protocol number, if an ACL is configured for a protocol not listed above

For TCP and UDP, a separate set of statistics is listed for each application port.

Clearing flow-based ACL statistics

To clear the ACL statistics, enter the following command at the Privileged EXEC level of the CLI.

```
ServerIronADX(config)# clear ip acl-traffic
```

Syntax: clear ip acl-traffic

Dropping all fragments that exactly match a flow-based ACL

For a packet fragment that is sent to the CPU for processing, the device compares the fragment's source and destination IP addresses against the interface's ACL entries. By default, if the fragment's source and destination IP addresses exactly match an ACL entry that also has Layer 4 information (source and destination TCP or UDP application ports), the device permits or denies the fragment according to the ACL.

On an individual interface basis, you can configure an IronCore device to automatically drop a fragment whose source and destination IP addresses exactly match an ACL entry that has Layer 4 information, even if that ACL entry's action is permit. To do so, enter the following command at the configuration level for an interface.

```
ServerIronADX(config-if-1/1)# ip access-group frag deny
```

Syntax: [no] ip access-group frag deny

Clearing the ACL statistics

Statistics on the ACL account report can be cleared:

- When a software reload occurs
- When the ACL is bound to or unbound from an interface
- When you enter the **clear access-list** command, as in the following example.

```
ServerIronADX(config)# clear access-list all
```

Syntax: clear access-list all | ethernet <*slot*>/<*port*>

Enter **all** to clear all statistics for all ACLs.

Use **ethernet** <*slot*>/<*port*> to clear statistics for ACLs a physical port.

Enabling ACL filtering of fragmented packets

Filtering fragmented packets for rule-based ACLs

By default, when a rule-based ACL is applied to a port, the port will use the ACL to permit or deny the first fragment of a fragmented packet, but forward subsequent fragments of the same packet in hardware. Generally, denying the first fragment of a packet is sufficient, since a transaction cannot be completed without the entire packet.

NOTE
The fragmentation support described in this section applies only to rule-based ACLs.

NOTE
Enhanced fragment handling is not supported on 10 Gigabit Ethernet modules. By default, 10 Gigabit Ethernet modules also forward the first fragment instead of using the ACLs to permit or deny the fragment.

For tighter control, you can enable CPU filtering of all packet fragments on a port. When you enable CPU filtering, the port sends all the fragments of a fragmented packet to the CPU. The CPU then permits or denies each fragment according to the ACL applied to the port. You can enable CPU filtering of fragments on individual ports.

You also can configure the port to drop all packet fragments.

To enable CPU filtering of packet fragments on an individual port, enter commands such as the following.

```
ServerIronADX(config)# interface ethernet 1/1
ServerIronADX(config-if-1/1)# ip access-group frag inspect
```

Syntax: [no] ip access-group frag inspect | deny

The **inspect | deny** parameter specifies whether you want fragments to be sent to the CPU or dropped:

- **inspect** – This option sends all fragments to the CPU.
- **deny** – This option begins dropping all fragments received by the port as soon as you enter the command. This option is especially useful if the port is receiving an unusually high rate of fragments, which can indicate a hacker attack.

Throttling the fragment rate

By default, when you enable CPU filtering of packet fragments, all fragments are sent to the CPU. Normally, the fragment rate in a typical network does not place enough additional load on the CPU to adversely affect performance. However, performance can be affected if the device receives a very high rate of fragments. For example, a misconfigured server or a hacker can affect the device's performance by flooding the CPU with fragments.

You can protect against fragment flooding by specifying the maximum number of fragments the device or an individual interface is allowed to send to the CPU in a one-second interval. If the device or an interface receives more than the specified number of fragments in a one-second interval, the device either drops or forwards subsequent fragments in hardware, depending on the action you specify. In addition, the device starts a holddown timer and continues to either drop or forward fragments until the holddown time expires.

The device also generates a Syslog message.

To specify the maximum fragment rate per second, enter commands such as the following.

```
ServerIronADX(config)# ip access-list frag-rate-on-system 15000 exceed-action
drop reset-interval 10
ServerIronADX(config)#ip access-list frag-rate-on-interface 5000 exceed-action
forward reset-interval 5
```

The first command sets the fragment threshold at 15,000 per second, for the entire device. If the device receives more than 15,000 packet fragments in a one-second interval, the device takes the specified action. The action specified with this command is to drop the excess fragments and continue dropping fragments for a holddown time of ten minutes. After the ten minutes have passed, the device starts sending fragments to the CPU again for processing.

The second command sets the fragment threshold at 5,000 for individual interfaces. If any interface on the device receives more than 5,000 fragments in a one-second interval, the device takes the specified action. In this case, the action is to forward the fragments in hardware without filtering them. The device continues forwarding fragments in hardware for five minutes before beginning to send fragments to the CPU again.

Both thresholds apply to the entire device. Thus, if an individual interface's fragment threshold is exceeded, the drop or forward action and the holddown time apply to all fragments received by the device.

Syntax: [no] ip access-list frag-rate-on-system <*num*> **exceed-action drop | forward reset-interval** <*mins*>

and

Syntax: [no] ip access-list frag-rate-on-interface <*num*> **exceed-action drop | forward reset-interval** <*mins*>

The <*num*> parameter specifies the maximum number of fragments the device or an individual interface can receive and send to the CPU in a one-second interval.

- **frag-rate-on-system** – Sets the threshold for the entire device. The device can send to the CPU only the number of fragments you specify per second, regardless of which interfaces the fragments come in on. If the threshold is exceeded, the device takes the exceed action you specify.
- **frag-rate-on-interface** – Sets the threshold for individual interfaces. If an individual interface receives more than the specified maximum number of fragments, the device takes the exceed action you specify.

The <*num*> parameter specifies the maximum number of fragments per second.

- For **frag-rate-on-system**, you can specify from 600 – 12800. The default is 6400.
- For **frag-rate-on-interface**, you can specify from 300 – 8000. The default is 4000.

The **drop | forward** parameter specifies the action to take if the threshold (<*num*> parameter) is exceeded:

- **drop** – fragments are dropped without filtering by the ACLs
- **forward** – fragments are forwarded in hardware without filtering by the ACLs

The <*mins*> parameter specifies the number of minutes the device will enforce the drop or forward action after a threshold has been exceeded. You can specify from 1 – 30 minutes, for **frag-rate-on-system** or **frag-rate-on-interface**.

Syslog messages for exceeded fragment thresholds

If a fragment threshold is exceeded, the device generates one of the following Syslog messages.

TABLE 4 Syslog messages for exceeded fragment threshold

Message level	Message	Explanation
Notification	ACL system fragment packet inspect rate <*rate*> exceeded	The <*rate*> indicates the maximum rate allowed.
Notification	ACL port fragment packet inspect rate <*rate*> exceeded on port <*portnum*>	The <*rate*> indicates the maximum rate allowed. The <*portnum*> indicates the port.

Enabling hardware filtering for packets denied by flow-based ACLs

By default, packets denied by ACLs are filtered by the CPU. You can enable the device to create CAM entries for packets denied by ACLs. This causes the filtering to occur in hardware instead of in the CPU.

When you enable hardware filtering of denied packets, the first time the device filters a packet denied by an ACL, the device sends the packet to the CPU for processing. The CPU also creates a CAM entry for the denied packet. Subsequent packets with the same address information are filtered using the CAM entry. The CAM entry ages out after two minutes if not used.

To enable hardware filtering of denied packets, enter the following command at the global CONFIG level of the CLI.

```
ServerIronADX(config)# hw-drop-acl-denied-packet
```

Syntax: [no] hw-drop-acl-denied-packet

Enabling strict TCP or UDP mode for flow-based ACLs

By default, when you use ACLs to filter TCP or UDP traffic, the Brocade device does not compare all TCP or UDP packets against the ACLs.

For TCP and UDP, the device first compares the source and destination information in a TCP control packet or a UDP packet against entries in the session table. The session table contains forwarding entries based on Layer 3 and Layer 4 information:

- If the session table contains a matching entry, the device forwards the packet, assuming that the first packet the device received with the same address information was permitted by the ACLs.
- If the session table does not contain a matching entry, the device sends the packet to the CPU, where the software compares the packet against the ACLs. If the ACLs permit the packet (explicitly by a permit ACL entry or implicitly by the absence of a deny ACL entry), the CPU creates a session table entry for the packet's forwarding information and forwards the packet.

For TCP, this behavior by default applies only to control packets, not to data packets. Control packets include packet types such as SYN (Synchronization) packets, FIN (Finish) packets, and RST (Reset) packets.

For tighter access or forwarding control, you can enable the device to perform strict TCP or UDP ACL processing. The following sections describe the strict modes in more detail.

Enabling strict TCP mode

By default, when you use ACLs to filter TCP traffic, the Brocade device does not compare all TCP packets against the ACLs. Instead, the device compares TCP control packets against the ACLs, but not data packets. Control packets include packet types such as SYN (Synchronization) packets, FIN (Finish) packets, and RST (Reset) packets.

In normal TCP operation, TCP data packets are present only if a TCP control session for the packets also is established. For example, data packets for a session never occur if the TCP SYN for that session is dropped. Therefore, by filtering the control packets, the Brocade device also implicitly filters the data packets associated with the control packets. This mode of filtering optimizes forwarding performance for TCP traffic by forwarding data packets without examining them. Since the data packets are present in normal TCP traffic only if a corresponding TCP control session is established, comparing the packets for the control session to the ACLs is sufficient for filtering the entire session including the data.

However, it is possible to generate TCP data packets without corresponding control packets, in test or research situations for example. In this case, the default ACL mode does not filter the data packets, since there is no corresponding control session to filter. To filter this type of TCP traffic, use the strict ACL TCP mode. This mode compares all TCP packets to the configured ACLs, regardless of whether the packets are control packets or data packets. If the ACLs permit the packet, the device creates a session entry for forwarding other TCP packets with the same Layer 3 and Layer 4 addresses.

NOTE
Regardless of whether the strict mode is enabled or disabled, the device always compares TCP control packets against the configured ACLs before creating a session entry for forwarding the traffic.

NOTE
If the device's configuration currently has ACLs associated with interfaces, remove the ACLs from the interfaces before changing the ACL mode.

To enable the strict ACL TCP mode, enter the following command at the global CONFIG level of the CLI.

```
ServerIronADX(config)# ip strict-acl-tcp
```

Syntax: [no] ip strict-acl-tcp

This command configures the device to compare all TCP packets against the configured ACLs before forwarding them.

To disable the strict ACL mode and return to the default ACL behavior, enter the following command.

```
ServerIronADX(config)# no ip strict-acl-tcp
```

NOTE
Enter the **ip rebind-acl** command at the global CONFIG level of the CLI to place the **ip strict-acl-tcp** or **no ip strict-acl-tcp** command into effect.

Enabling strict UDP mode

By default, when you use ACLs to filter UDP traffic, the Brocade device does not compare all UDP packets against the ACLs. Instead, the device compares the source and destination information against entries in the session table. The session table contains forwarding entries based on Layer 3 and Layer 4 information:

- If the session table contains a matching entry, the device forwards the packet, assuming that the first packet the device received that contains the same address information was permitted by the ACLs.
- If the session table does not contain a matching entry, the device sends the packet to the CPU, where the software compares the packet against the ACLs. If the ACLs permit the packet (explicitly by a permit ACL entry or implicitly by the absence of a deny ACL entry), the CPU creates a session table entry for the packet's forwarding information and forwards the packet.

For tighter control, the software provides the strict ACL UDP mode. When you enable strict UDP processing, the device sends every UDP packet to the CPU and compares the packet against the configured ACLs.

NOTE
If the device's configuration currently has ACLs associated with interfaces, remove the ACLs from the interfaces before changing the ACL mode.

To enable the strict ACL UDP mode, enter the following command at the global CONFIG level of the CLI.

```
ServerIronADX(config)# ip strict-acl-udp
```

Syntax: [no] ip strict-acl-udp

This command configures the device to compare all UDP packets against the configured ACLs before forwarding them.

To disable the strict ACL mode and return to the default ACL behavior, enter the following command.

```
ServerIronADX(config)# no ip strict-acl-udp
```

NOTE
Enter the **ip rebind-acl** command at the global CONFIG level of the CLI to place the **ip strict-acl-udp** or **no ip strict-acl-udp** command into effect.

Configuring ACL packet and flow counters

You can configure counters for packets and flows that match entries in an ACL. Using the CLI, you can display the contents of the counters and clear them:

- The ACL packet counter feature provides an accurate count of packets matching individual ACL entries.
- The ACL flow counter feature provides an approximate count of flows matching individual ACL entries. This feature can be used for troubleshooting purposes to provide an indication of flow activity against an ACL. Each time the Brocade device receives the first packet of a flow matching an entry in an ACL list, the flow counter for that ACL entry is incremented by one. If a flow lasts longer than two minutes, the flow counter for the ACL entry is incremented again.

  NOTE
  The ACL flow counter feature is designed to monitor the general volume of flow activity for an ACL. It is not intended to be used for accounting purposes.

The ACL flow and packet counters are incremented differently depending on whether packets are handled by the Management Processor (MP), and whether they are permit or deny flows.

The Management Processor (MP) handles flows as follows.

For flows handled by the Management Processor:

- For permit flows, only flows are counted. If a permit flow lasts longer than two minutes, the flow counter is incremented again.
- For deny flows, only packets are counted.

By default the ACL packet and flow counters are disabled. To activate them, enter the following command.

```
ServerIronADX(config)# enable-acl-counter
```

Syntax: [no] enable-acl-counter

Once the ACL packet and flow counters are enabled, you can disable them with the **no** form of the **enable-acl-counter** command. Disabling and then re-enabling the ACL packet and flow counters resets them to zero.

To display the packet and flow counters for ACL 100.

```
ServerIronADX# show access-list 100
Extended IP access list 100 (Total flows: 432, Total packets: 42000)
  permit tcp 1.1.1.0 0.0.0.255 any (Flows: 80, Packets: 12900)
  deny udp 1.1.1.0 0.0.0.255 any (Flows: 121, Packets: 20100)
  permit ip 2.2.2.0 0.0.0.255 any (Flows: 231, Packets: 9000)
```

Syntax: show access-list <*acl-num*> **|** <*acl-name*> **| all**

To clear the flow counters for ACL 100.

```
ServerIronADX# clear access-list 100
```

Syntax: clear access-list <*acl-num*> **|** <*acl-name*> **| all**

ACLs and ICMP

This section describes how ACLs can be used to filter traffic based on ICMP packets.

Using flow-based ACLs to filter ICMP packets based on the IP packet length

To configure an extended ACL that filters based on the IP packet length of ICMP packets, enter commands such as the following.

```
ServerIronADX(config)#access-list 105 deny icmp any any echo ip-pkt-len 92
ServerIronADX(config)#access-list 105 deny icmp any any echo ip-pkt-len 100
ServerIronADX(config)#access-list 105 permit ip any any
```

The commands in this example deny (drop) ICMP echo request packets that contain a total length of 92 or 100 in the IP header field. You can specify an IP packet length of 1 – 65535. Refer to the section "ICMP filtering with flow-based ACLs" on page 79 for additional information on using ICMP to filter packets.

ICMP filtering with flow-based ACLs

Most Brocade software releases that support flow-based ACLs filter traffic based on the following ICMP message types:

- echo
- echo-reply
- information-request
- mask-reply
- mask-request
- parameter-problem
- redirect
- source-quench
- time-exceeded
- timestamp-reply
- timestamp-request
- unreachable

- *<num>*

Also, to create ACL policies that filter ICMP message types, you can either enter the description of the message type or enter its type and code IDs. Furthermore ICMP message type filtering is now available for rule-based ACLs on BigIron Layer 2 Switch and Layer 3 Switch images.

Numbered ACLs

For example, to deny the echo message type in a numbered ACL, enter commands such as the following when configuring a numbered ACL.

```
ServerIronADX(config)# access-list 109 deny ICMP any any echo
```

or

```
ServerIronADX(config)# access-list 109 deny ICMP any any 8 0
```

Syntax: **[no] access-list** *<num>*

Syntax: **deny | permit icmp** *<source-ip-address>* **|** *<source-ip-address/subnet-mask>* **| any | host** *<source-host>*
<destination-ip-address> **|** *<destination-ip-address/subnet-mask>* **| any | host** *<destination-host>*
<icmp-type> **|** *<icmp-type-number>* *<icmp-code-number>*

The **deny | permit** parameter indicates whether packets that match the policy are dropped or forwarded.

You can either enter the name of the message type for *<icmp-type>* or the type number and code number of the message type. Refer to Table 5 on page 81 for valid values.

Named ACLs

For example, to deny the administratively-prohibited message type in a named ACL, enter commands such as the following.

```
ServerIronADX(config)# ip access-list extended melon
ServerIronADX(config-ext-nacl)# deny ICMP any any administratively-prohibited
```

or

```
ServerIronADX(config)# ip access-list extended melon
ServerIronADX(config-ext-nacl)# deny ICMP any any 3 13
```

Syntax: **[no] ip access-list extended** *<acl-num>* **|** *<acl-name>*

Syntax: **deny | permit icmp** *<source-ip-address>* **|** *<source-ip-address/subnet-mask>* **| any | host** *<source-host>*
<destination-ip-address> **| destination-ip-address/subnet-mask> | any | host** *<destination-host>*
<icmp-type> **|** *<icmp-type-number>* *<icmp-code-number>*

The **extended** parameter indicates the ACL entry is an extended ACL.

The *<acl-name>* | *<acl-num>* parameter allows you to specify an ACL name or number. If using a name, specify a string of up to 256 alphanumeric characters. You can use blanks in the ACL name if you enclose the name in quotation marks (for example, “ACL for Net1”). The *<acl-num>* parameter allows you to specify an ACL number if you prefer. If you specify a number, enter a number from 100 – 199 for extended ACLs.

The **deny | permit** parameter indicates whether packets that match the policy are dropped or forwarded.

You can either use the *<icmp-type>* and enter the name of the message type or use the *<icmp-type-number>* *<icmp-ode-number>* parameter and enter the type number and code number of the message. Refer to Table 5 for valid values.

NOTE
"X" in the Type-Number or Code-Number column in Table 5 means the device filters any traffic of that ICMP message type.

TABLE 5 ICMP message types and codes

ICMP message type	Type	Code
administratively-prohibited	3	13
any-icmp-type	x	x
destination-host-prohibited	3	10
destination-host-unknown	3	7
destination-net-prohibited	3	9
destination-network-unknown	3	6
echo	8	0
echo-reply	0	0
general-parameter-problem **NOTE:** This message type indicates that required option is missing.	12	1
host-precedence-violation	3	14
host-redirect	5	1
host-tos-redirect	5	3
host-tos-unreachable	3	12
host-unreachable	3	1
information-request	15	0
log		
mask-reply	18	0
mask-request	17	0
net-redirect	5	0
net-tos-redirect	5	2
net-tos-unreachable	3	11
net-unreachable	3	0
packet-too-big	3	4
parameter-problem **NOTE:** This message includes all parameter problems	12	0
port-unreachable	3	3
precedence-cutoff	3	15

TABLE 5 ICMP message types and codes

ICMP message type	Type	Code
protocol-unreachable	3	2
reassembly-timeout	11	1
redirect **NOTE:** This includes all redirects.	5	x
router-advertisement	9	0
router-solicitation	10	0
source-host-isolated	3	8
source-quench	4	0
source-route-failed	3	5
time-exceeded	11	x
timestamp-reply	14	0
timestamp-request	13	0
ttl-exceeded	11	0
unreachable **NOTE:** This includes all unreachable messages	3	x

Using ACLs and NAT on the same interface (flow-based ACLs)

You can use ACLs and NAT on the same interface, as long as you follow these guidelines:

- You must use the **ip strict-acl-tcp** command when configuring ACLs and NAT is configured on the same Layer 2 Switch. (Refer to the instructions below on how to use this command.)
- Do not enable NAT on an interface until you have applied ACLs (as described below) to the interface. If NAT is already enabled, you must disable it, apply the ACLs, then re-enable NAT on the interface.
- Enable the strict TCP mode.
- On the inside NAT interface (the one connected to the private addresses), apply inbound ACLs that permit TCP, UDP, and ICMP traffic to enter the device from the private sub-net.

You can use a standard ACL to permit all traffic (including TCP, UDP, and ICMP traffic) or an extended ACL with separate entries to explicitly permit TCP, UDP, and ICMP traffic.

NOTE
You do not need to apply ACLs to permit TCP, UDP, and ICMP traffic unless you are applying other ACLs to the interface as well. If you do not plan to apply any ACLs to a NAT interface, then you do not need to apply the ACLs to permit TCP, UDP, and ICMP traffic.

Here is an example of how to configure device to use ACLs and NAT on the same interfaces. In this example, the inside NAT interface is port 1/1 and the outside NAT interface is port 2/2.

The following commands enable the strict TCP mode and configure an ACL to permit all traffic from the 10.10.200.x sub-net. A second ACL denies traffic from a specific host on the Internet.

```
ServerIronADX(config)# ip strict-acl-tcp
ServerIronADX(config)# access-list 1 permit 10.10.200.0 0.0.0.255
ServerIronADX(config)# access-list 2 deny 209.157.2.184
```

The following commands configure global NAT parameters.

```
ServerIronADX(config)# ip nat inside source list 1 pool outadds overload
ServerIronADX(config)# ip nat pool outadds 204.168.2.1 204.168.2.254 netmask
255.255.255.0
```

The following commands configure the inside and outside NAT interfaces. Notice that the ACLs are applied to the inbound direction on the inside NAT interface, and are applied ***before*** NAT is enabled. In this example, ACL 1 permits all traffic to come into the inside interface from the private sub-net. ACL 2 denies traffic from a specific host from going out the interface to the private sub-net.

```
ServerIronADX(config)# interface ethernet 1/1
ServerIronADX(config-if-1/1)# ip address 10.10.200.1 255.255.255.0
ServerIronADX(config-if-1/1)# ip access-group 1 in
ServerIronADX(config-if-1/1)# ip access-group 2 out
ServerIronADX(config-if-1/1)# ip nat inside
ServerIronADX(config-if-1/1)# interface ethernet 2/2
ServerIronADX(config-if-2/2)# ip address 204.168.2.78 255.255.255.0
ServerIronADX(config-if-2/2)# ip nat outside
```

NOTE
Enter the **ip rebind-acl** command at the global CONFIG level of the CLI to place the **ip strict-acl-tcp** command into effect.

Displaying ACL bindings

You can display which ACLs (IPv4 and IPv6) are bound to which interfaces as shown in the following.

```
ServerIronADX# show access-list bindings
Access-list binding configuration:
!
interface ethernet 2
ip access-group 2 in
ipv6 traffic-filter acl1 in
!
interface ve 2
ip access-group 111 in
ipv6 traffic-filter acl2 out
```

Syntax: show access-list bindings

Troubleshooting rule-based ACLs

Use the following methods to troubleshoot a rule-based ACL:

- To display the number of Layer 4 CAM entries being used by each ACL, enter the **show access-list** <*acl-num*> | <*acl-name*> | **all** command. Refer to "Displaying the number of Layer 4 CAM entries" on page 53.

- To view the types of packets being received on an interface, enable ACL statistics using the **enable-acl-counter** command, reapply the ACLs by entering the **ip rebind-acl all** command, then display the statistics by entering the **show ip acl-traffic** command.
- To determine whether an ACL entry is correctly matching packets, add the **log** option to the ACL entry, then reapply the ACL. This forces the device to send packets that match the ACL entry to the CPU for processing. The **log** option also generates a Syslog entry for packets that are permitted or denied by the ACL entry.
- To determine whether the issue is specific to fragmentation, remove the Layer 4 information (TCP or UDP application ports) from the ACL, then reapply the ACL.

If you are using another feature that requires ACLs, either use the same ACL entries for filtering and for the other feature, or change to flow-based ACLs.

Chapter

IPv6 Access Control Lists 3

IACL overview

ServerIron ADX supports IPv6 Access Control Lists (ACLs) in hardware. The maximum number of ACL entries you can configure is a system-wide parameter and depends on the device you are configuring. You can configure up to the maximum number of 1024 entries in any combination in different ACLs. The total number of entries in all ACLs cannot exceed the system maximum of 1024

By default, IPv6 ACLs are processed in hardware and all IPv6 ACL rules are stored in TCAM.

An IPv6 ACL is composed of one or more conditional statements that pose an action (permit or deny) if a packet matches a specified source or destination prefix. There can be up to 1024 IPv6 ACL statements per device. When the maximum number of IPv6 ACL rules are reached, the following error message will display on the console:

```
IPv6 Hardware ACL rules cannot be configured,exceeds the maximum hardware limit of
1024 entries
Insufficient hardware resource for binding the ACL scale1 to interface Port or
Slot/Port.
```

In ACLs with multiple statements, you can specify a priority for each statement.The specified priority determines the order in which the statement appears in the ACL. The last statement in each IPv6 ACL is an implicit deny statement for all packets that do not match the previous statements in the ACL.

You can configure an IPv6 ACL on a global basis, then apply it to the incoming IPv6 packets on specified interfaces. You can apply only one IPv6 ACL to an interface's incoming traffic. When an interface receives an IPv6 packet, it applies the statement within the ACL in their order of appearance to the packet. As soon as a match occurs, the ServerIron ADX takes the specified action (permit or deny the packet) and stops further comparison for that packet.

Brocade's IPv6 ACLs enable traffic filtering based on the following information:

- IPv6 protocol
- Source IPv6 address
- Destination IPv6 address
- Source TCP or UDP port (if the IPv6 protocol is TCP or UDP)
- Destination TCP or UDP port (if the IPv6 protocol is TCP or UDP)

The IPv6 protocol can be one of the following well-known names or any IPv6 protocol number from 0 – 255:

- Authentication Header (AHP)
- Encapsulating Security Payload (ESP)
- Internet Control Message Protocol (ICMP)
- Internet Protocol Version 6 (IPv6)
- Stream Control Transmission Protocol (SCTP)

- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)

NOTE
TCP and UDP filters will be matched only if they are listed as the first option in the extension header.

For TCP and UDP, you also can specify a comparison operator and port name or number. For example, you can configure a policy to block web access to a specific website by denying all TCP port 80 (HTTP) packets from a specified source IPv6 address to the website's IPv6 address.

This chapter contains the following sections:

- "Configuring an IPv6 ACL" on page 87
- "Applying an IPv6 ACL to an interface" on page 93
- "Displaying ACLs" on page 94

Configuration Notes

- Either IPv6 must be enabled globally or an IPV6 address must be configured on an interface before IPv6 ACLs can be configured.
- An IPv6 ACL can include up to 1024 entries or statements.
- Only named ACLs are supported.
- Only Inbound ACLs are supported.
- If an IPv6 ACL has the implicit **deny** condition, make sure it also **permits** the IPv6 link-local address, in addition to the global unicast address. Otherwise, routing protocols such as OSPF will not work. To view the link-local address, use the **show ipv6 interface** command.
- You cannot disable IPv6 on an interface to which an ACL is bound. Attempting to do so will cause the system to return the following error message.

```
ServerIronADX(config-if-e1000-7)#no ipv6 enable
Error:  Port 7 has IPv6 ACL configured.  Cannot disable IPv6
```

To disable IPv6, first remove the ACL from the interface.

Processing of IPv6 ACLs

There are two ways that IPv6 ACLs are processed in Brocade devices: in software and in hardware. This processing differs depending on the software release that you are running. These differences are described in the following sections.

Prior to release 12.3.01

Prior to release 12.3.01, IPv6 ACLs were processed as described in the following:

For deny and permit actions:

All permit and deny packets are forwarded to the BPs and the BPs perform the ACL processing.

Beginning with release 12.3.01 and later

Beginning with release 12.3.01, IPv6 ACLs are processed as described in the following:

For deny actions:

All deny packets are dropped in hardware.

For permit actions:

For all traffic, packets are processed in hardware and then forwarded to the BPs. The BPs do not take any action on the ACLs.

Backwards compatibility option:

You can use the **ipv6 flow-based-acl-enable** command to provide backwards compatibility for IPv6 ACL processing. If this command is configured, packets are processed in hardware and then forwarded to the BPs where the BPs also process the ACLs. This command is configured as shown in the following.

```
ServerIronADX(config)# ipv6 flow-based-acl-enable
```

Syntax: ipv6 flow-based-acl-enable

Configuring an IPv6 ACL

To configure an IPv6 ACL, do the following:

1. Create the IPv6 ACL.
2. Apply the IPv6 ACL to the interface.

Example Configurations

To configure an access list that blocks all Telnet traffic received on port 1/1 from IPv6 host 2000:2382:e0bb::2, enter the following commands.

```
ServerIronADX(config)# ipv6 access-list fdry
ServerIronADX(config-ipv6-access-list-fdry)# deny tcp host 2000:2382:e0bb:
:2 any eq telnet
ServerIronADX(config-ipv6-access-list-fdry)# permit ipv6 any any
ServerIronADX(config-ipv6-access-list-fdry)# exit
ServerIronADX(config)# int eth 1/1
ServerIronADX(config-if-1/1)# ipv6 traffic-filter fdry in
ServerIronADX(config)# write memory
```

Here is another example of commands for configuring an ACL and applying it to an interface.

```
ServerIronADX(config)# ipv6 access-list netw
ServerIronADX(config-ipv6-access-list-netw)# permit icmp 2000:2383:
e0bb::/64 2001:3782::/64
ServerIronADX(config-ipv6-access-list-netw)# deny ipv6 host 2000:2383:
e0ac::2 host 2000:2383:e0aa:0::24
ServerIronADX(config-ipv6-access-list-netw)# deny udp any any
ServerIronADX(config-ipv6-access-list-netw)# permit ipv6 any any
```

The first condition permits ICMP traffic from hosts in the 2000:2383:e0bb::x network to hosts in the 2001:3782::x network.

The second condition denies all IPv6 traffic from host 2000:2383:e0ac::2 to host 2000:2383:e0aa:0::24.

The third condition denies all UDP traffic.

The fourth condition permits all packets that are not explicitly denied by the other entries. Without this entry, the ACL would deny all incoming IPv6 traffic on the ports to which you assigned the ACL.

The following commands apply the ACL "netw" to the incoming traffic on port 1/2 and to the incoming traffic on port 4/3.

```
ServerIronADX(config)# int eth 1/2
ServerIronADX(config-if-1/2)# ipv6 traffic-filter netw in
ServerIronADX(config-if-1/2)# exit
ServerIronADX(config)# int eth 4/3
ServerIronADX(config-if-4/3)# ipv6 traffic-filter netw in
ServerIronADX(config)# write memory
```

Here is another example:

```
ServerIronADX(config)# ipv6 access-list nextone
ServerIronADX(config-ipv6-access-list rtr)# deny tcp 2001:1570:21::/24
2001:1570:22::/24
ServerIronADX(config-ipv6-access-list rtr)# deny udp any range 5 6
2001:1570:22::/24
ServerIronADX(config-ipv6-access-list rtr)# permit ipv6 any any
ServerIronADX(config-ipv6-access-list rtr)# write memory
```

The first condition in this ACL denies TCP traffic from the 2001:1570:21::x network to the 2001:1570:22::x network.

The next condition denies UDP packets from any source with source UDP port in ranges 5 to 6 and whose destination is to the 2001:1570:22::/24 network.

The third condition permits all packets containing source and destination addresses that are not explicitly denied by the first two. Without this entry, the ACL would deny all incoming IPv6 traffic on the ports to which you assign the ACL.

A **show running-config** command displays the following:

```
ServerIronADX(config)# show running-config
ipv6 access-list rtr
deny tcp 2001:1570:21::/24 2001:1570:22::/24
deny udp any range 5 6 2001:1570:22::/24
permit ipv6 any any
```

A **show ipv6 access-list** command displays the following:

```
ServerIronADX(config)# sh ipv6 access-list rtr
ipv6 access-list rtr: 3 entries
 deny tcp 2001:1570:21::/24 2001:1570:22::/24
 deny udp any range 5 6 2001:1570:22::/24
 permit ipv6 any any
```

The following commands apply the ACL “rtr” to the incoming traffic on ports 2/1 and 2/2.

```
ServerIronADX(config)# int eth 2/1
ServerIronADX(config-if-2/1)# ipv6 traffic-filter rtr in
ServerIronADX(config-if-2/1)# exit
ServerIronADX(config)# int eth 2/2
ServerIronADX(config-if-2/2)# ipv6 traffic-filter rtr in
ServerIronADX(config)# write memory
```

Default and Implicit IPv6 ACL Action

The default action when no IPv6 ACLs are configured on an interface is to permit all IPv6 traffic. However, once you configure an IPv6 ACL and apply it to an interface, the default action for that interface is to deny all IPv6 traffic that is not explicitly permitted on the interface.

- If you want to tightly control access, configure ACLs consisting of permit entries for the access you want to permit. The ACLs implicitly deny all other access.
- If you want to secure access in environments with many users, you might want to configure ACLs that consist of explicit deny entries, then add an entry to permit all access to the end of each ACL. The permit entry permits packets that are not denied by the deny entries.

Every IPv6 ACL has the following implicit condition as its last match conditions:

deny ipv6 any any – Denies IPv6 traffic. You must enter a **permit ipv6 any any** as the last statement in the access-list if you want to permit IPv6 traffic that were not denied by the previous statements.

NOTE
If an IPv6 ACL has the implicit **deny** condition, make sure it also **permits** the IPv6 link-local address, in addition to the global unicast address. Otherwise, routing protocols such as OSPF will not work. To view the link-local address, use the **show ipv6 interface** command.

The conditions are applied in the order shown above, with **deny ipv6 any any** as the last condition applied.

For example, if you want to deny ICMP neighbor discovery acknowledgement, then permit any remaining IPv6 traffic, enter commands such as the following:

```
ServerIronADX(config)# ipv6 access-list netw
ServerIronADX(config-ipv6-access-list-netw)# permit icmp 2000:2383:e0bb:
:/64 2001:3782::/64
ServerIronADX(config-ipv6-access-list-netw)# permit ipv6 any any
```

The first permit statement permits ICMP traffic from hosts in the 2000:2383:e0bb::x network to hosts in the 2001:3782::x network.

The deny statement denies ICMP neighbor discovery acknowledgement.

The last entry permits all packets that are not explicitly denied by the other entries. Without this entry, the ACL will deny all incoming IPv6 traffic on the ports to which you assigned the ACL.

Furthermore, if you add the statement **deny icmp any any** in the access list, then all neighbor discovery messages will be denied. You must explicitly enter the **permit icmp any any nd-na** and **permit icmp any any nd-ns** statements just before the **deny icmp** statement if you want the ACLs to permit neighbor discovery as in the example below.

```
ServerIronADX(config)# ipv6 access-list netw
ServerIronADX(config-ipv6-access-list-netw)# permit icmp 2000:2383:e0bb:
:/64 2001:3782::/64
ServerIronADX(config-ipv6-access-list-netw)# deny icmp any any
ServerIronADX(config-ipv6-access-list-netw)# permit ipv6 any any
```

ACL Syntax

NOTES: The following features are not supported:

- **ipv6-operator flow-label**
- **ipv6-operator fragments** when any protocol is specified. The option "fragments" can be specified only when "permit/deny ipv6" is specified. If you specify "tcp" or any other protocol instead of "ipv6" the keyword, "fragments" cannot be used.
- **ipv6-operator routing** when any protocol is specified. (Same limitation as for **ipv6-operator fragments**)

- The following **ICMPv6 Message Types** are not supported:

DECIMAL	<0-255> ICMP message type
beyond-scope	Destination Unreachable ICMP message, Beyond Scope
destination-unreachable	Destination Unreachable ICMP messages
dscp	Match dscp value in IPv6 packet
echo-reply	Echo Reply ICMP message
echo-request	Echo Request ICMP message
header	Parameter Problem ICMP Message,Header Error
hop-limit	Time Exceeded ICMP Message,In Transit
log	Log matches against this entry
mld-query	MLD Query Message
mld-reduction	MLD Reduction Message
mld-report	MLD Report Message
nd-na	ND Neighbor Advertisement Message
nd-ns	ND Neighbor Solicitation Message
next-header	Parameter Problem ICMP Message,Next Header
no-admin	Destination Unreachable ICMP Message, Administratively Prohibited
no-route	Destination Unreachable ICMP Message, No Route
packet-too-big	Packet Too Big ICMP Message
parameter-option	Parameter Problem ICMP Message,Option
parameter-problem	Parameter Problem ICMP Messages
port-unreachable	Destination Unreachable ICMP Message, Port Unreachable
reassembly-timeout	Time Exceeded ICMP Message, ReassemblyTimeout
renum-command	Renumber Command ICMP Message
renum-result	Renumber Result ICMP Message
renum-seq-number	Renumber Sequence Number ICMP Message
router-advertisement	Router Advertisement ICMP Message
router-renumbering	All Router Renumbering ICMP Messages
router-solicitation	Router Solicitation ICMP Message
sequence	Sequence number for this entry
time-exceeded	Time Exceeded ICMP Messages
traffic-policy	Attach traffic policy by name
unreachable	Destination Unreachable ICMP message,Address Unreachable

When creating ACLs, use the appropriate syntax below for the protocol you are filtering.

For IPv6 and Supported Protocols Other than ICMP, TCP, or UDP
Syntax: [no] ipv6 access-list *<acl-name>*

Syntax: **permit | deny** <*protocol*>
<*ipv6-source-prefix/prefix-length*> **| any | host** <*source-ipv6_address*>
<*ipv6-destination-prefix/prefix-length*> **| any | host** <*ipv6-destination-address*>
[ipv6-operator [<*value*>]] **[log]**

For ICMP

Syntax: **[no] ipv6 access-list** <*acl-name*>

Syntax: **permit | deny icmp** <*ipv6-source-prefix/prefix-length*> **| any | host**
<*source-ipv6_address*>
<*ipv6-destination-prefix/prefix-length*> **| any | host** <*ipv6-destination-address*>
[ipv6-operator [<*value*>]]
[[<*icmp-type*>][<*icmp-code*>]] | [<*icmp-message*>] **[log]**

For TCP

Syntax: **[no] ipv6 access-list** <*acl-name*>

Syntax: **permit | deny** <*tcp*>
<*ipv6-source-prefix/prefix-length*> **| any | host** <*source-ipv6_address*> **[tcp-udp-operator [source-port-number]]**
<*ipv6-destination-prefix/prefix-length*> **| any | host** <*ipv6-destination-address*>
[tcp-udp-operator [destination-port- number]]
[ipv6-operator [<value>]] **[log]**

For UDP

Syntax: **[no] ipv6 access-list** <*acl-name*>

Syntax: **permit | deny** <*udp*>
<*ipv6-source-prefix/prefix-length*> **| any | host** <*source-ipv6_address*> **[tcp-udp-operator [source port number]]**
<*ipv6-destination-prefix/prefix-length*> **| any | host** <*ipv6-destination-address*>
[tcp-udp-operator [destination port number]]
[ipv6-operator [<*value*>]] **[log]**

TABLE 6 Syntax Descriptions

Arguments...	Description...
ipv6 access-list <acl-name>	Enables the IPv6 configuration level and defines the name of the IPv6 ACL. The <acl-name> can contain up to 199 characters and numbers, but cannot begin with a number and cannot contain any spaces or quotation marks.
permit	The ACL will permit (forward) packets that match a policy in the access list.
deny	The ACL will deny (drop) packets that match a policy in the access list.
icmp	Indicates the you are filtering ICMP packets.
protocol	The type of IPv6 packet you are filtering. You can specify a well-known name for some protocols whose number is less than 255. For other protocols, you must enter the number. Enter "?" instead of a protocol to list the well-known names recognized by the CLI. IPv6 protocols include: AHP – Authentication Header ESP – Encapsulating Security Payload IPv6 – Internet Protocol version 6 SCTP – Stream Control Transmission Protocol

TABLE 6 Syntax Descriptions

Arguments...	Description...
<ipv6-source-prefix>/<prefix-length>	The <ipv6-source-prefix>/<prefix-length> parameter specify a source prefix and prefix length that a packet must match for the specified action (deny or permit) to occur. You must specify the <ipv6-source-prefix> parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373. You must specify the <prefix-length> parameter as a decimal value. A slash mark (/) must follow the <ipv6-prefix> parameter and precede the <prefix-length> parameter.
<ipv6-destination-prefix>/<prefix-length>	The <ipv6-destination-prefix>/<prefix-length> parameter specify a destination prefix and prefix length that a packet must match for the specified action (deny or permit) to occur. You must specify the <ipv6-destination-prefix> parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373. You must specify the <prefix-length> parameter as a decimal value. A slash mark (/) must follow the <ipv6-prefix> parameter and precede the <prefix-length> parameter
any	When specified instead of the <ipv6-source-prefix>/<prefix-length> or <ipv6-destination-prefix>/<prefix-length> parameters, matches any IPv6 prefix and is equivalent to the IPv6 prefix::/0.
host	Allows you specify a host IPv6 address. When you use this parameter, you do not need to specify the prefix length. A prefix length of all128 is implied.
icmp-message	ICMP packets are filtered by ICMP messages. See the "Configuring IPv6 ICMP Features" section of the "Configuring IPv6 Connectivity" chapter of the *ServerIron ADX TrafficWorks Switching and Routing Guide*.
tcp	Indicates the you are filtering TCP packets.
udp	Indicates the you are filtering UDP packets.
<ipv6-source-prefix>/<prefix-length>	The <ipv6-source-prefix>/<prefix-length> parameter specify a source prefix and prefix length that a packet must match for the specified action (deny or permit) to occur. You must specify the <ipv6-source-prefix> parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373. You must specify the <prefix-length> parameter as a decimal value. A slash mark (/) must follow the <ipv6-prefix> parameter and precede the <prefix-length> parameter.
<ipv6-destination-prefix>/<prefix-length>	The <ipv6-destination-prefix>/<prefix-length> parameter specify a destination prefix and prefix length that a packet must match for the specified action (deny or permit) to occur. You must specify the <ipv6-destination-prefix> parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373. You must specify the <prefix-length> parameter as a decimal value. A slash mark (/) must follow the <ipv6-prefix> parameter and precede the <prefix-length> parameter
any	When specified instead of the <ipv6-source-prefix>/<prefix-length> or <ipv6-destination-prefix>/<prefix-length> parameters, matches any IPv6 prefix and is equivalent to the IPv6 prefix::/0.
host	Allows you specify a host IPv6 address. When you use this parameter, you do not need to specify the prefix length. A prefix length of all128 is implied.

TABLE 6 Syntax Descriptions

Arguments...	Description...
tcp-udp-operator	The <tcp-udp-operator> parameter can be one of the following: **eq** – The policy applies to the TCP or UDP port name or number you enter after **eq**. **gt** – The policy applies to TCP or UDP port numbers greater than the port number or the numeric equivalent of the port name you enter after **gt**. Enter "?" to list the port names. **lt** – The policy applies to TCP or UDP port numbers that are less than the port number or the numeric equivalent of the port name you enter after **lt**. **neq** – The policy applies to all TCP or UDP port numbers except the port number or port name you enter after **neq**. **range** – The policy applies to all TCP port numbers that are between the first TCP or UDP port name or number and the second one you enter following the **range** parameter. The range includes the port names or numbers you enter. For example, to apply the policy to all ports between and including 23 (Telnet) and 53 (DNS), enter the following: **range 23 53**. The first port number in the range must be lower than the last number in the range. The <source-port number> and <destination-port-number> for the tcp-udp-operator is the number of the port.
ipv6-operator	Allows you to filter the packets further by using one of the following options: • **dscp** – The policy applies to packets that match the traffic class value in the traffic class field of the IPv6 packet header. This operator allows you to filter traffic based on TOS or IP precedence. You can specify a value from 0 – 63. • **fragments** – The policy applies to fragmented packets that contain a non-zero fragment offset. **NOTE:** This option is not applicable to filtering based on source or destination port, TCP flags, and ICMP flags. • **routing** – The policy applies only to IPv6 source-routed packets. **NOTE:** This option is not applicable to filtering based on source or destination port, TCP flags, and ICMP flags.
log	Allows statistics that match the ACL statement to be entered in the Syslog.

Applying an IPv6 ACL to an interface

To apply an IPv6 ACL to an interface, enter commands such as the following:

```
ServerIronADX(config)# interface ethernet 3/1
ServerIronADX(config-if-e100-3/1)# ipv6 traffic-filter access1 in
```

This example applies the IPv6 ACL “access1” to incoming IPv6 packets on Ethernet interface 3/1. As a result, Ethernet interface 3/1 denies all incoming packets from the site-local prefix fec0:0:0:2::/64 and the global prefix 2001:100:1::/48 and permits all other incoming packets.

Syntax: ipv6 traffic-filter <*ipv6-acl-name*> **in**

For the <*ipv6-acl-name*> parameter, specify the name of an IPv6 ACL created using the **ipv6 access-list** command.

The **in** keyword applies the specified IPv6 ACL to incoming IPv6 packets on the interface.

Displaying ACLs

To display the ACLs configured on a device, enter the **show ipv6 access-list** command. Here is an example:

```
ServerIronADX# show ipv6 access-list
ipv6 access-list v6-acl1: 1 entries
 deny ipv6 any any
ipv6 access-list v6-acl2: 1 entries
 permit ipv6 any any
ipv6 access-list v6-acl3: 2 entries
 deny ipv6 2001:aa:10::/64 any
 permit ipv6 any any
ipv6 access-list v6-acl4: 2 entries
 deny ipv6 2002:aa::/64 any
 permit ipv6 any any
ipv6 access-list v6-acl5: 6 entries
 permit tcp 2002:bb::/64 any
 permit ipv6 2002:bb::/64 any
 permit ipv6 2001:aa:101::/64 any
 permit ipv6 2001:aa:10::/64 2001:aa:102::/64
 permit ipv6 host 2001:aa:10::102 host 2001:aa:101::102
 permit ipv6 any any fragments
```

Syntax: show ipv6 access-list [<*access-list-name*>]

Displaying ACLs bound to an interface

To display ACLs bound to an interface, enter the **show access-list bindings** command. Here is an example:

```
ServerIronADX# show access-list bindings
Access-list binding configuration:
!
interface ethernet 1
ipv6 traffic-filter ipv61 in
!
interface ethernet 2
ipv6 traffic-filter icmp_any in
!
ServerIronADX 1000#
```

Syntax: show access-list bindings

Using an ACL to Restrict SSH Access

To configure an ACL that restricts SSH access to an IPv6 device, first create the named ACL with the ACL statements. Then use the **ssh access-group** command to restrich SSH access for IPv6:

```
ServerIronADX(config)# ipv6 access-list test2
ServerIronADX(config-ipv6-access-list test2)#  deny ipv6  host 2000:1::1 any log
ServerIronADX(config-ipv6-access-list test2)#  permit ipv6  2000:1::0/32 any
ServerIronADX(config-ipv6-access-list test2)#  permit ipv6  2000:2::0/32 any
ServerIronADX(config-ipv6-access-list test2)#  permit ipv6 host 2000:3::1 any
ServerIronADX(config-ipv6-access-list test2)# exit
ServerIronADX(config)# ssh access-group ipv6 test2
```

Syntax: [no] ssh access-group ipv6 <*acl-name*>

Using an ACL to Restrict Telnet Access

To configure an ACL that restricts Telnet access to an IPv6 device, first create the named ACL with the ACL statements. Then use the **telnet access-group** command to restrict Telnet access for IPv6:

```
ServerIronADX(config)# ipv6 access-list test1
ServerIronADX(config-ipv6-access-list test1)#  deny ipv6  host 2000:1::1 any log
ServerIronADX(config-ipv6-access-list test1)#  permit ipv6  2000:1::0/32 any
ServerIronADX(config-ipv6-access-list test1)#  permit ipv6  2000:2::0/32 any
ServerIronADX(config-ipv6-access-list test1)#  permit ipv6 host 2000:3::1 any
ServerIronADX(config-ipv6-access-list test1)# exit
ServerIronADX(config)# telnet access-group ipv6 test1
```

Syntax: telnet access-group ipv6 <*acl-name*>

Logging IPv6 ACLs

Logging for IPv6 ACLs is disabled by default. To enable logging, enable it for each IPv6 ACL, then include the logging option in an ACL statement. Logging at both levels need to be configured in order for statistics for packets that match the condition to be logged. For example:

```
ServerIronADX(config)# ipv6 access-list acl2
ServerIronADX(config-ipv6-access-list-acl2)# logging-enable
ServerIronADX(config-ipv6-access-list-acl2)# permit tcp host
2002:200:12d:1300:204:23ff:fec7:dabf any eq http
ServerIronADX(config-ipv6-access-list-acl2)# deny icmp 2002:200:12d:1300::/64 any
echo-reply log
ServerIronADX(config-ipv6-access-list-acl2)# permit ipv6 any any
```

Syntax: [no] logging-enable

NOTE
Syntax for the **log** option in an IPv6 ACL statement are presented in the section "ACL Syntax" on page 89.

NOTE
Permit logging is not currently supported.

3

Chapter

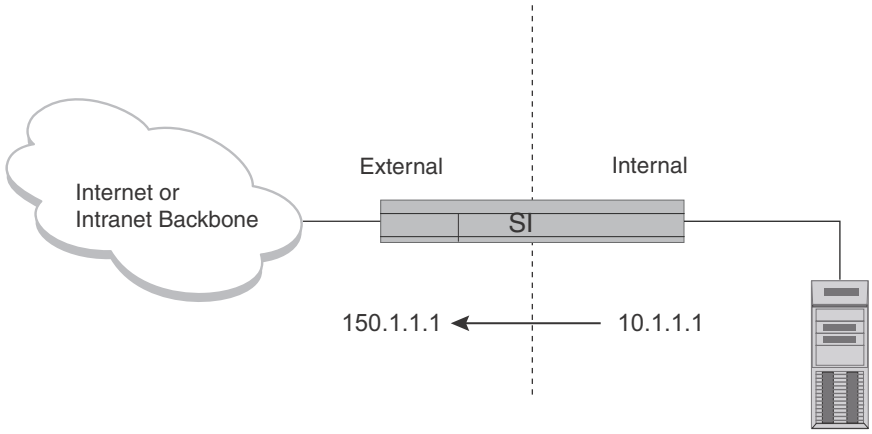
4

Network Address Translation

Introduction

Network Address Translation (NAT) translates one IP address into another. For example, it translates an internal private IP address (nonregistered) into an external unique IP address (registered) used on the Internet.

FIGURE 5 Mapping an internal address to an external address



NAT also provides a more graceful renumbering strategy for organizations changing service providers or voluntarily renumbering into Classless Interdomain Routing (CIDR) blocks.

The standard NAT support described in this section provides translation for hosts attached to private networks on the ServerIron ADX, and is separate from the virtual IP address features provided for Server Load Balancing (SLB). For example, standard NAT is not related to source IP addresses used for multinetting the ServerIron ADX, performing health checks on remote servers, and so on.

Configuring NAT

The following types of NAT are supported:

- **Static NAT** — Maps a specific global IP address (Internet IP address) with a specific private address. Static translation ensures the software always maps the same public address to a given private address. For example, you can map 10.1.1.1 to 150.1.1.1. Use static NAT when you want a specific host in the private network to always use the same Internet address when communicating outside the private network. ServerIron ADX supports both inside to outside network translation and outside to inside network Nat translation.

- **Dynamic NAT** — Maps private addresses to Internet addresses. The Internet addresses come from a pool of addresses that you configure. For example, you can dynamically translate the global pool 150.1.1.10 - 19 to private pool 10.1.1.1 - 254. In Figure 6, the pool is the range of addresses from 209.157.1.2/24 – 209.157.1.254/24. With dynamic NAT, the software uses a round robin technique to select a global IP address to map to a private address from a pool you configure.

  Dynamic NAT uses Port Address Translation (PAT). Otherwise, the return traffic cannot be reliably de-multiplexed to the correct internal client.

NOTE
You can configure both dynamic and static NAT on the same device. When you configure both types of NAT, static NAT takes precedence over dynamic NAT. Thus, if you configure a static NAT translation for a private address, the ServerIron ADX always uses that translation instead of creating a dynamic one.

Configuring static NAT

Use the **ip nat inside source static** command to explicitly map a private address to an Internet address. Static NAT ensures a specific host in the private network is always mapped to the Internet address you specify.

To map a private address 10.10.10.69 to an Internet address 209.157.1.69, enter the command such as the following.

```
ServerIronADX(config)# ip nat inside source static 10.10.10.69 209.157.1.69
```

Syntax: [no] ip nat inside source static <*private-ip*> <*global-ip*> [<*priority*>] **list** [<*acl-id*>]

The <*private-ip*> variable specifies the private IP address.

The <*global-ip*> variable specifies the IP address. The ServerIron ADX supports up to 255 global IP addresses.

The <*priority*> variable specifies a value of 1 or 2 and enables static NAT redundancy. A value of 2 means higher priority, and will be the owner of the NAT IP as long as the system is up.

The **list** parameter specifies the access list identified by the <*acl-id*> variable that will permit only the configured tcp or udp port numbers.

Configuring dynamic NAT

To configure dynamic NAT, perform the following tasks:

- Configure a standard or extended ACL for each private address range for which you want to provide NAT.

  NOTE
  Named ACLS are not supported with NAT. You must use a numbered ACL.

- Configure a pool for each consecutive range of Internet addresses to which you want NAT to be able to map the private addresses specified in the ACLs. Each pool must contain a range with no gaps. If your Internet address space has gaps, configure separate pools for each consecutive range within the address space.
- Associate a range of private addresses (specified in a standard or extended ACL) with a pool.

Configuring an address pool

Use the **ip nat pool** command to configure the address pool. For an example, refer to "Dynamic NAT configuration example 1" on page 100.

Syntax: **[no] ip nat pool** <*pool-name*> <*start-ip*> <*end-ip*> **netmask** <*ip-mask*> **| prefix-length** <*length*> **| port-pool-range** <*priority-value*>

The <*pool-name*> parameter specifies the name assigned to the pool. It can be up to 255 characters long and can contain special characters and internal blanks. If you use internal blanks, you must use quotation marks around the entire name.

The <*start-ip*> parameter specifies the IP address at the beginning of the pool range. Specify the lowest-numbered IP address in the range.

The <*end-ip*> parameter specifies the IP address at the end of the pool range. Specify the highest-numbered IP address in the range.

NOTE
The address range cannot contain any gaps. Make sure you own all the IP addresses in the range. If the range contains gaps, you must create separate pools containing only the addresses you own.

The **netmask** <*ip-mask*> | prefix-length <*length*> parameter specifies a classical sub-net mask (example: netmask 255.255.255.0) or the length of a CIDR prefix (example: prefix-length 24). The ServerIron ADX supports up to 255 global IP addresses.

The **port-pool-range** <*priority-value*> parameter enables dynamic NAT redundancy, where the <*priority-value*> can be 1 or 2. A range value of 2 indicates higher priority for the NAT IP. A 2 value also means the source ports allocated for the NAT IP are from the higher range.

Associating a range of private addresses with a pool and enabling PAT

Use **ip nat inside source list** to associate a private address range with a pool of Internet addresses and enable PAT. For an example, refer to "Dynamic NAT configuration example 1" on page 100.

Syntax: **[no] ip nat inside source list** <*acl-id*> **pool** <*pool-name*>

The **inside source** keyword specifies that the translation applies to private addresses sending traffic to the Internet (inside source).

The **list** <*acl-id*> parameter specifies a standard or extended ACL. Named ACLS are not supported with NAT. You must use a numbered ACL.

The **pool** <*pool-name*> parameter specifies the pool name. You must create the pool before you can use it with this command.

NAT configuration examples

The following sections provide both Dynamic and Static NAT configuration examples.

NOTE
A ServerIron ADX can have a maximum of 255 global IP addresses, in a single pool or multiple pools.

Dynamic NAT configuration example 1

This section describes the Dynamic NAT configuration shown in Figure 6.

FIGURE 6 Minimum required commands

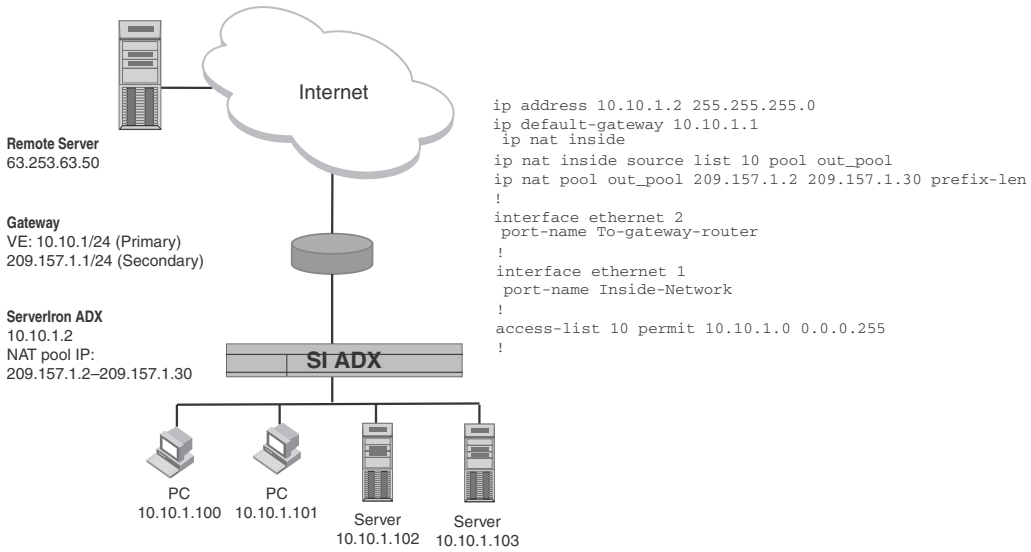


Figure 6 shows an example of a network using dynamic NAT on a ServerIron ADX ADX. The device is acting as a gateway to connect a private network to the Internet. The private network, which can also be considered as the **inside** network, is using IP addresses in the range of 10.10.1.2 - 10.10.1.254 with a 24-bit subnet mask.

The ServerIron ADX is connected to the Internet through a router. The **outside** interface of the ServerIron ADX has a global IP address of 209.157.1.1. The ServerIron ADX also has a pool of global IP addresses, which are used to map internal IP addresses.

Minimum required commands for dynamic NAT configuration.

1. Identify an internal and external interface on the ServerIron ADX. In this example, Ethernet 1/2 and 1/1 are used.

   ```
   int eth 1/2
   int eth 1/1
   ```

2. Assign IP addresses to the interfaces and define the outside and inside boundaries of the NAT mechanism.

   ```
   ServerIronADX(config)# int eth 1/2
   ServerIronADX(config-if-1/2)# ip address 209.157.1.1/24
   ServerIronADX(config-if-1/2)# ip nat outside
   ServerIronADX(config-if-1/2)# int eth 1/1
   ServerIronADX(config-if-1/1)# ip address 10.10.10.1/24
   ServerIronADX(config-if-1/1)# ip nat inside
   ```

   On Switch (S) code, enable NAT globally.

   ```
   ServerIronADX(config)# ip nat inside
   ```

   On Router (R) code, enable NAT on interfaces (both **ip nat inside** and **outside** should be enabled). The interfaces can also be physical interfaces (not necessarily virtual interfaces).

```
ServerIronADX(config-ve-2)#ip nat inside
ServerIronADX(config-ve-3)#ip nat outside
```

3. Configure a numbered ACL and permit the IP addresses on the inside. Then define the global address pool and enable dynamic NAT.

```
ServerIronADX(config)# access-list 101 permit ip 10.10.1.0/24 any
ServerIronADX(config)# ip nat pool global_pool 209.157.1.2 209.157.1.254
prefix-length 24
```

   Make sure you specify **permit** in the ACL, rather than **deny**. If you specify **deny**, the ServerIron ADX will not provide NAT for the addresses.

4. Tie the inside source list to the global pool and enable PAT (overload) to send traffic out the external interface.

```
ServerIronADX(config)# ip nat inside source list 101 pool global_pool
```

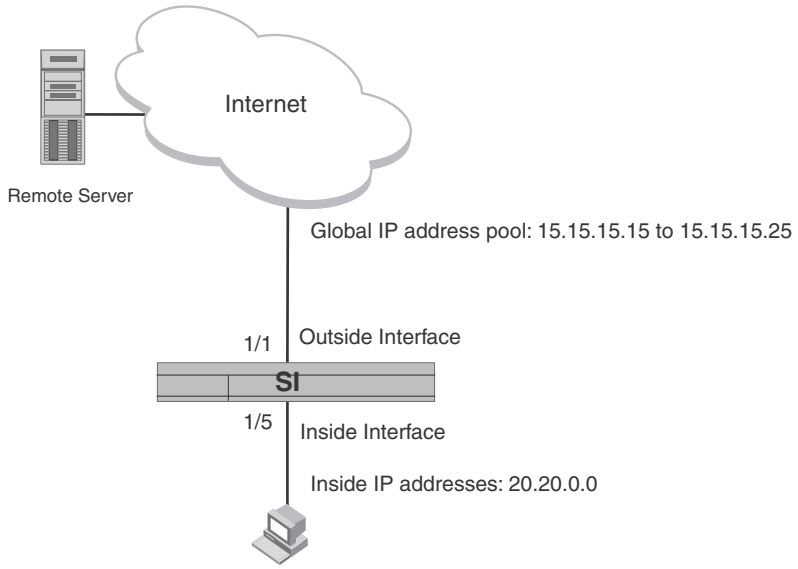
5. **rconsole** into the BP and verify the translation is working correctly.

```
rconsole x/x
show ip nat statistic
show ip nat translation
```

Dynamic NAT configuration example 2

In the following example, the ServerIron ADX is configured to translate inside hosts in the 20.20.0.0 network to unique global addresses in the 15.15.15.15/24 network.

FIGURE 7 Example of a dynamic NAT configuration - translating inside host addresses to unique pool addresses



This example requires that interfaces 1/5 and 1/1 be configured as Inside and Outside interfaces respectively as shown.

```
ServerIronADX(config)# interface ethernet 1/5
ServerIronADX(config-if-e1000-1/5) ip address 20.20.50.1 255.255.0.0
ServerIronADX(config-if-e1000-1/5) ip nat inside
```

```
ServerIronADX(config)# interface ethernet 1/1
ServerIronADX(config-if-e1000-1/5) ip address 30.30.0.1 255.255.0.0
ServerIronADX(config-if-e1000-1/5) ip nat outside
```

The following command creates a pool of IP NAT addresses from 15.15.15.15 to 15.15.15.25 named p1.

```
ServerIronADX(config)# ip nat pool p1 15.15.15.15 15.15.15.25 prefix-len 24
```

An ACL is created to permit traffic from inside hosts in the 20.20.0.0 network as shown.

```
ServerIronADX(config)# access-list 1 permit 20.20.0.0 0.0.255.255
```

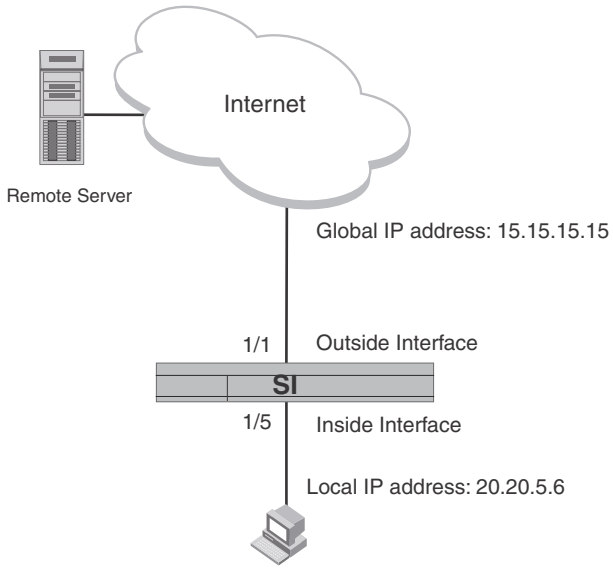
The following command ties the inside source list defined in ACL “1” to the pool named “p1” and enables PAT to send traffic out the interface defined as “outside”.

```
ServerIronADX(config)# ip nat inside source list 1 pool p1
```

Static NAT configuration example

The following examples describe how to configure a Static NAT configuration for Inside to Outside and Outside to Inside translation for the example shown in Figure 8.

FIGURE 8 Example of a static NAT configuration using router code



Configured for inside to outside translation

In the following example, the ServerIron ADX is configured to translate the local host IP address 20.20.5.6 to the unique global address 15.15.15.15.

This example requires that Interfaces 1/5 and 1/1 be configured as Inside and Outside interfaces respectively as shown.

```
ServerIronADX(config)# interface ethernet 1/5
ServerIronADX(config-if-e1000-1/5) ip address 20.20.50.1 255.255.0.0
ServerIronADX(config-if-e1000-1/5) ip nat inside
```

```
ServerIronADX(config)# interface ethernet 1/1
ServerIronADX(config-if-e1000-1/5) ip address 30.30.0.1 255.255.0.0
ServerIronADX(config-if-e1000-1/5) ip nat outside
```

The following command configures the ServerIron ADX to translate IP packets with a local IP address of 20.20.5.6 to the global IP address 15.15.15.15.

```
ServerIronADX(config)# ip nat inside source static 20.20.5.6 15.15.15.15
```

Configured for outside to inside translation

To configure the network shown in Figure 8 for Outside to Inside translation the only requirement is that the Interface configured as an Outside interface must be configured with an additional IP address in the 15.15.15.0/24 network as shown in the following.

```
ServerIronADX(config)# interface ethernet 1/1
ServerIronADX(config-if-e1000-1/5) ip address 30.30.0.1 255.255.0.0
ServerIronADX(config-if-e1000-1/5) ip address 15.15.15.100 255.255.0.0
ServerIronADX(config-if-e1000-1/5) ip nat outside
```

PAT

Dynamic NAT uses Port Address Translation (PAT). Since there is no one-to-one mapping between private addresses and global addresses, PAT maps a client's IP address and TCP/UDP port to both a global IP address and a TCP/UDP port. In this way, the ServerIron ADX can map many private addresses to the same public address and use TCP/UDP ports to uniquely identify the private hosts.

PAT maps a client's IP address and TCP or UDP port number to both an IP address and a TCP or UDP port number. In this way, the ServerIron ADX can map many private addresses to the same public address and use TCP or UDP port numbers to uniquely identify the private hosts.

NOTE
PAT is also called **overloading** an inside global address.

Example

Inside address	Outside address
10.10.10.2:6000	209.157.1.2:1024
10.10.10.3:6000	209.157.1.2:1025
10.10.10.4:6000	209.157.1.2:1026

NAT is mapping the same global IP address to three different private addresses along with their TCP or UDP ports, but uses a different TCP or UDP port number for each private address to distinguish them. Notice that the PAT feature does not attempt to use the same TCP or UDP port number as in the client's packet.

Forwarding packets without NAT translation

When ServerIron ADX receives a non-SYN packet for a TCP flow from an internal NAT client and no sessions are found, then by default ServerIron drops that packet. Optionally, you can forward that packets without NAT translation by entering the following command.

```
ServerIronADX(config)# nat-forward-no-session
```

Syntax: [no] nat-forward-no-session

Translation timeouts

The NAT translation table contains all the currently active NAT translation entries on the device. An active entry is one the ServerIron ADX creates for a private address when the client at that address sends traffic.

NAT performs the following steps to provide an address translation for a source IP address:

- NAT looks in the translation table for an active NAT entry for the translation. If the table contains an active entry for the session, the ServerIron ADX uses that entry.
- If NAT does not find an active entry in the NAT translation table, NAT creates an entry and places the entry in the table. The entry remains in the table until the entry times out.

Each NAT entry remains in the translation table until the entry ages out.

After the configuration of a NAT timeout the following occurs:

- Existing entries of the modified protocol remain in the translation table.
- If existing session entries send or receive packets then the timeout value is updated to the new configured value.
- If existing session entries do not process any traffic they will continue to age out in accordance with the old timeout value.
- When the timeout (age of the session) expires, forward and reverse sessions in the ServerIron ADX are deleted with no further actions. If traffic is received on this flow the ServerIron ADX drops the packets because it will not find sessions related to that particular flow.

Configuring the NAT translation aging timer

Use the **ip nat translation** command to alter the NAT translation aging timer.

The NAT translation table contains all the currently active NAT translation entries on the device. An active entry is one that the ServerIron ADX created for a private address when that client at that address sent traffic to the Internet. NAT performs the following steps to provide an address translation for a source IP address:

- The feature looks in the NAT translation table for an active NAT entry for the translation. If the table contains an active entry for the session, the ServerIron ADX uses that entry.
- If NAT does not find an active entry in the NAT translation table, NAT creates an entry and places the entry in the table. The entry remains in the table until the entry times out.

```
ServerIronADX(config)# ip nat translation tcp-timeout 1800
```

Syntax: [no] **ip nat translation dns-timeout | finrst-timeout | icmp-timeout | syn-timeout | tcp-timeout | udp-timeout** <*secs*> **maximum**

Syntax: [no] **ip nat translation max-entries** <*number-of-entries*>

The **dns-timeout** keyword indicates connections to a Domain Name Server (DNS). The default is 120 seconds.

The **finrst-timeout** keyword identifies TCP FIN (finish) and RST (reset) packets, which normally terminate TCP connections. The default is 120 seconds. This timer is not related to **tcp-timeout**, which applies to packets to or from a host address that is mapped to an global IP address and a TCP port number (PAT feature). The **finrst-timeout** applies to packets that terminate a TCP session, regardless of the host address or whether PAT is used.

The **icmp-timeout** keyword indicates timeout for NAT ICMP flows

The **syn-timeout** keyword indicates timeout for NAT TCP flows after a SYN

The **tcp-timeout** keyword indicates dynamic entries that use PAT based on TCP port numbers. The default is 120 seconds. This timer applies only to TCP sessions that do not end “gracefully”, with a TCP FIN or TCP RST.

The **udp-timeout** keyword indicates dynamic entries that use PAT based on UDP port numbers. The default is 120 seconds.

The <*secs*> parameter specifies number of seconds, 0– 3600. Use **maximum** to set the maximum timeout value. For example, 3,600 seconds.

The **max-entries** <*number-of-entries*> parameter specifies the maximum number of NAT entries

Stateless static IP NAT

A ServerIron ADX creates sessions for Static NAT by default. You can prevent a ServerIron ADX from creating sessions for static NAT traffic with the following command.

```
ServerIronADX(config)# [no] ip nat stateless
```

Syntax: ip nat stateless

For “ip nat stateless“ to work, the existing command, “ip nat inside source static” must already be configured.

Example

```
ip nat inside source static 10.45.16.103 10.45.16.10
```

NOTE
FTP, RTSP and other similar complex protocols are not supported. The traffic applicable for IP NAT Stateless are TCP, UDP, and ICMP.

NOTE
You must reload a ServerIron ADX whenever changes are made to a running IP NAT configuration.

Redundancy

The IP NAT Redundancy feature implements a separate protocol to negotiate IP address ownership of NAT IP addresses.

The new protocol is similar to the symmetric VIP protocol and uses any L2 link to exchange the NAT PDUs. Both ServerIronADXs will run a “symmetric VIP like” protocol to report and receive ownership (similar to the VLAN AD protocol in symmetric SLB). When one ServerIron ADX goes down, the peer ServerIron ADX will become the master for that NAT IP (in case of static NAT) or NAT pool (in case of dynamic NAT). However, the NAT IP/NAT pool ownership is used only to decide which ServerIronADX responds to the ARP request for the NAT IP. Both ServerIronADXs are allowed to use the NAT IP in keeping with the design for symmetric VIP (sym-active SLB).

The global **ip policy** dependency is as follows:

- SLB — not needed
- IP NAT — not needed
- TCS — An **ip policy** must be defined. Without it, caching will not work.

Enabling IP NAT

When a ServerIron ADX is configured with Switch code, NAT is enabled globally but when it is configured with Router Code, it is enabled per-interface.

NOTE
ServerIron ADX ADX does not support IP NAT inside and outside on the same physical interface.

Enabling IP NAT globally

The following command enables IP NAT globally.

```
ServerIronADX(config)# ip nat inside
```

Syntax: [no] ip nat inside

Enabling IP NAT per-interface

When enabled per-interface, IP NAT must be enabled exclusively “inside” or “outside” on a physical or virtual interface as shown in the following example.

```
ServerIronADX(config)# interface ethernet 1/5
ServerIronADX(config-if-e1000-1/5) ip nat inside
```

Syntax: [no] ip nat [inside | outside]

The **inside** parameter configures the interface as an IP NAT inside interface.

The **outside** parameter configures the interface as an IP NAT outside interface.

Enabling static NAT redundancy

To enable static NAT redundancy, enter the **ip nat inside source static** command such as the following.

```
ServerIronADX(config)# ip nat inside source static 10.10.10.10 63.32.23.1 2
```

Syntax: ip nat inside source static <*ip-addr1*> <*ip-addr2*> <*priority-value*>

The existing **ip nat inside** command has been extended to include a <*priority-value*>, which is used to determine the owner of the NAT IP address.

The <*priority-value*> can be 1 or 2. 2 is the higher priority, and will be the owner of the NAT IP as long as the system is up.

Enabling dynamic NAT redundancy

To enable dynamic NAT redundancy, enter commands such as the following.

```
ServerIronADX(config)# ip nat pool foo 63.23.1.2 63.23.1.4 prefix 24
ServerIronADX(config)# ip nat pool foo port-pool-range 2
```

Syntax: ip nat pool <*pool-name*> **port-pool-range** <*priority-value*>

The **port-pool-range** <*priority-value*> parameter supports redundancy for IP NAT pool addresses. This parameter is similar to the priority value for static NAT, except it also determines the range of source ports allocated by the NAT IP (which prevents source port collision).

In ServerIron ADX, the **ip nat pool** <name> **port-pool-range** command is mandatory for running router code in HA setups. This command decides the ownership of the IP NAT pool and, when using router code, this command has to be used in tandem with **ip nat pool** <name> <Start-IP-address> <End-IP-address> command.

The <*priority-value*> can be 1 or 2. A range value of 2 indicates higher priority for the NAT IP. It also means the source ports allocated for the NAT IP are from the higher range.

NOTE
A distribution of port ranges is not required for static NAT, as it does not involve PAT.

Displaying NAT information

The following sections describe how to display NAT information.

Displaying NAT statistics

To display NAT statistics, enter commands such as the following.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1#ServerIronADX_Lower1/1# show ip nat stat
Debug counters:
 TCP FWD:
send nat unreachable tcp fwd) =0 nat tcp no ports avl = 2867811
nat tcp status zero = 0 nat tcp ip status zero = 0
nat tcp usr index null = 0
 TCP REV:
send nat unreachable (tcp rev) =0 nat tcp rev no ports avl = 0
nat tcp rev status zero = 0 nat tcp rev ip status zero = 0
nat tcp rev usr index null = 0
 UDP FWD:
send nat unreachable (udp_fwd) =0 nat udp fwd no ports avl = 0
nat udp fwd status zero = 0 nat udp fwd ip status zero = 0
nat udp fwd usr index null = 0
 UDP REV:
send nat unreachable (udp rev) =0 nat udp rev no ports avl = 0
nat udp rev status zero = 0 nat udp rev ip status zero = 0
nat udp rev usr index null = 0
nat corruption = 0 rtsp port unavailable = 0
RTSP inside alloc same = 53271 RTSP reply port not same= 0
Wrong port range = 0
Total translations: 12433 (0 static, 12433 dynamic)
Hits: 5786108 Misses: 94
Expired translations: 5773769
Dynamic mappings:
 pool p1: prefix_len= 24
     start 15.15.15.15 end 15.15.15.20
     total addresses 6 overloaded
[0]: h: 0 t: 0 m: 248a6000 T: 384 f: 384
[1]: h: 0 t: 0 m: 248ab000 T: 384 f: 384
[2]: h: 0 t: 0 m: 29f33000 T: 384 f: 384
[3]: h: 0 t: 0 m: 2a5cb000 T: 384 f: 384
[4]: h: 0 t: 0 m: 2a5d4000 T: 384 f: 384
[5]: h: 0 t: 0 m: 2a5dd000 T: 384 f: 384
[0]: h: 0 t: 0 m: 248a8000 T: 320 f: 320
[1]: h: 0 t: 0 m: 29f30000 T: 320 f: 320
[2]: h: 0 t: 0 m: 2a5c4000 T: 320 f: 320
[3]: h: 0 t: 0 m: 2a5cd000 T: 320 f: 320
[4]: h: 0 t: 0 m: 2a5d6000 T: 320 f: 320
[5]: h: 0 t: 0 m: 2a5df000 T: 320 f: 320
 [0]: h: 972 t: 5653 m: 248a2000 T: 7168 f: 4681
 [1]: h: 972 t: 5653 m: 2a5bc000 T: 7168 f: 4681
 [2]: h: 2520 t: 656 m: 2a5c0000 T: 7168 f: 5304
 [3]: h: 2520 t: 655 m: 2a5c7000 T: 7168 f: 5303

Sess: Total 2000000, Avail 1950226, NAT 24886
Stream media=1, RTSP=(1:94547), MMS=(1:0), PNM=(1:0)
Inside global      Last Inside Local  xmit pkts  xmit bytes rx pkts    rx bytes   cnt
15.15.15.15        20.20.2.15         89909      7619664    44954      8023584    2487
15.15.15.16        20.20.2.16         89907      7619447    44950      8023575    2487
15.15.15.17        20.20.2.11         67428      5714424    33712      6017592    1864
15.15.15.18        20.20.2.12         67428      5714424    33712      6017592    1865
15.15.15.19        20.20.2.13         67432      5714763    33714      6017949    1865
15.15.15.20        20.20.2.14         67434      5714883    33714      6017949    1865
```

Syntax: show ip nat statistics

TABLE 7 Display fields for show ip nat statistics

This field...	Displays...
send nat unreachable (tcp fwd)	Indicates the number of times that a “port unreachable” message was generated for NAT TCP forward traffic.
nat tcp no ports avl	Indicates the number of times that a “port unreachable” message was generated because the ServerIron could not get a port from the port pool for a NAT IP for TCP forward traffic.
nat tcp status zero	Indicates the number of times that an error in NAT translation for TCP forward traffic.
nat tcp ip status zero	Indicates errors in NAT translation for TCP forward traffic.
nat tcp usr index null	Indicates the number of times that a “port unreachable” message was generated because the ServerIron could not create a user session for TCP forward traffic.
send nat unreachable (tcp rev)	Indicates the number of times that a “port unreachable” message was generated for TCP reverse traffic.
nat tcp rev no ports avl	Indicates the number of times that a “port unreachable” message was generated because the ServerIron could not get a port from the port pool for an IP NAT for TCP reverse traffic.
nat tcp rev status zero	Indicates the number of times that an error in NAT translation for TCP reverse traffic has occurred.
nat tcp rev ip status zero	Indicates the number of times that an error in NAT translation for TCP reverse traffic has occurred.
nat tcp rev usr index null	Indicates the number of times that a “port unreachable” message was generated because the ServerIron could not create a user session for TCP reverse traffic.
send nat unreachable (udp fwd)	Indicates the number of times that a “port unreachable” message was generated for UDP forward traffic.
nat udp fwd no ports avl	Indicates the number of times that a “port unreachable” message was generated because the ServerIron could not get a port from the port pool for a NAT IP for UD forward traffic.
nat udp fwd status zero	Indicates the number of times that an error in NAT translation for UDP forward traffic has occurred.
nat udp fwd status zero	Indicates the number of times that an error in NAT translation for UDP forward traffic has occurred.
nat udp fwd usr index null	Indicates the number of times that a “port unreachable” message was generated because the ServerIron could not create a user session for UDP forward traffic.
send nat unreachable (udp rev)	Indicates the number of times that a “port unreachable” message was generated for UDP reverse traffic because the Serveriron could not get a port from the port pool for a NAT IP.
nat udp rev no port avl	Indicates the number of times that a “port unreachable” message was generated because the ServerIron could not get a port from the port pool for a NAT IP for UDP reverse traffic.
nat udp rev status zero	Indicates the number of times that an error in NAT translation for UDP reverse traffic has occurred.

TABLE 7 Display fields for show ip nat statistics (Continued)

This field...	Displays...
nat udp rev ip status zero	Indicates the number of times that an error in NAT translation for UDP reverse traffic has occurred.
nat udp rev usr index null	Indicates the number of times that a “port unreachable” message was generated because the ServerIron could not create a a user session for UDP reverse traffic.
sw l4 nat corruption	Indicates the number of instances of NAT session corruption.
rstp port unavailable	Indicates the number of times that a NAT port was not available for RSTP.
RTSP inside alloc same	Indicates the number of times that the used port and proposed client port were the same for RSTP.
RTSP reply port not same	Indicates the number of times that the used port and proposed client port were not the same for RTSP.
Wrong port range	Indicates the number of times that the NAT port used a port in the wrong port range. For example, where a NAT port used a port from the normal port pool range for RTSP.
Port Pool Parameters	
[x]	The variable represented by "x" represents the index of the IP address in the IP NAT pool. For example, [0] refers to the first IP address in the IP pool (216:220:209:230). [1] refers to the second IP address in this IP pool (216:220:209:231).
h	The value following "h:" refers to the head of the port pool for the IP address in the IP NAT pool. The head indicates the location in the port pool where the next port will be allocated from.
t	The value following "t:" refers to the tail of the port pool for the IP address in the IP NAT pool. The tail indicates the location in the port pool where the next port will be freed from.
T	The value following "T:" refers to the total number of ports in the port pool for that IP address in the IP NAT pool.
f	The value following "f:" refers to the number of free ports in the port pool for this IP address.

Displaying NAT translation

To display the currently active NAT translations, enter the following command.

```
ServerIronADX(1/1)# show ip nat translation

Pro Inside global       Inside local       Outside local    Outside global
tcp 10.1.1.92:11021       5.1.1.2:32784        10.1.1.1:23       10.1.1.1:23
```

Syntax: show ip nat translation

NOTE
You can enter this command only when you **rconsole** in to a BP. The command is not supported on the Main Processor CPU.

TABLE 8 Display fields for show ip nat translation

This field...	Displays...
Pro	When PAT is enabled, this field indicates the protocol NAT is using to uniquely identify the host. NAT can map the same IP address to multiple hosts and use the protocol port to distinguish among the hosts. This field can have one of the following values: • **tcp** – In addition to this IP address, NAT is associating a TCP port with the host on the private network. • **udp** – In addition to this IP address, NAT is associating a UDP port with the host on the private network.
Inside global	The Internet address mapped to the private address listed in the Inside local field for inside NAT.
Inside local	The private address mapped to the Internet private address listed in the Inside global field for inside NAT.
Outside global	The destination of the traffic. If PAT is enabled, the TCP or UDP port also is shown. **NOTE:** Outside NAT is not supported.
Outside local	The destination of the traffic. If PAT is enabled, the TCP or UDP port also is shown. **NOTE:** Outside NAT is not supported.

Displaying NAT redundancy information

You can display information about the state of the static NAT IP or NAT pool (dynamic), the MAC address used, and the configured priority. The MAC address used for the NAT IP is a special construct, where the last 3 bytes of the MAC address are derived from the shared NAT IP address (similar to the symmetric MAC).

To display NAT redundancy information, enter the following command.

```
ServerIronADX# show ip nat redundancy (on active)
NAT Pool Start IP: 10.1.1.150 Mac address: 020c.db01.0196
State: Active Priority: High
NAT Pool Start IP: 10.1.1.91   Mac address: 020c.db01.015b
State: Active Priority: High
NAT Pool Start IP: 10.1.1.92   Mac address: 020c.db01.015c
State: Active Priority: High
NAT Pool Start IP: 10.1.1.95   Mac address: 020c.db01.015f
State: Active Priority: High
```

The two "**Priority**" options are "**High**" and "**Low**". That is, 2 or 1.

The two "**State**" options are "**Active**" and "**standby**".

```
ServerIronADX# show ip nat redundancy (on standby)
NAT Pool Start IP: 10.1.1.150 Mac address: 020c.db01.0196
State: Standby Priority: Low
Standby Idle count: 0 Threshold: 20
NAT Pool Start IP: 10.1.1.91   Mac address: 020c.db01.015b
State: Standby Priority: Low
Standby Idle count: 0 Threshold: 20
NAT Pool Start IP: 10.1.1.92   Mac address: 020c.db01.015c
State: Standby Priority: Low
Standby Idle count: 0 Threshold: 20
NAT Pool Start IP: 10.1.1.95   Mac address: 020c.db01.015f
State: Standby Priority: Low
Standby Idle count: 0 Threshold: 20
```

Syntax: show ip nat redundancy

Displaying VRRPE information

To display VRRPE information, enter the following command.

```
ServerIronADX_Lower# show ip vrrp-e brief

Total number of VRRP-Extended routers defined: 2
Interface VRID CurPri P State  Master addr   Backup addr       VIP

 v5         1    125 P Master Local         Unknown         5.1.1.9
 v10        2    125 P Master Local         Unknown         10.1.1.9
```

Syntax: show ip vrrp-e brief

Clearing NAT entries from the table

Use the **clear ip nat** command to manually clear entries from the NAT table.

Syntax: clear ip nat <*protocol*> **inside** <*global-ip*> <*global-port*> <*private-ip*> <*local-port*>

The <*protocol*> parameter specifies the protocol type and can be **tcp** or **udp** plus its global or local port number.

To clear a specific NAT entry based on the private and global IP addresses, enter the command such as the following.

```
ServerIronADX# clear ip nat inside 209.157.1.43 10.10.10.5
```

This command clears the inside NAT entry that maps private address 10.10.10.5 to Internet address 209.157.1.43.

Syntax: clear ip nat inside <*global-ip*> <*private-ip*>

To clear all static and dynamic entries from the NAT translation table, enter the following command.

```
ServerIronADX# clear ip nat all
```

Syntax: clear ip nat all

Chapter

Syn-Proxy and DoS Protection 5

This chapter describes how to configure Syn-Proxy and DOS protection features on the ServerIron ADX Traffic Managers.

Understanding Syn-Proxy

Syn-Proxy™ allows TCP connections to be terminated on the ServerIron ADX. When Syn-Proxy is enabled, the ServerIron ADX completes the three-way handshake with a connecting client. Only when the three-way handshake is completed does the ServerIron ADX establish a connection with the destination server and forward packets from the client to the server.

In a TCP SYN attack, the attacker floods a host with TCP SYN packets. The host replies with SYN-ACK packets, but the attacker does not send the ACK packet. The handshake remains incomplete, and the host goes into a perpetual wait-state for it to be completed. As a result, the resources available for TCP connections are rapidly depleted and the host is unable to accept any further TCP connections.

ServerIron ADX prevents these types of attacks by sitting in between the host and attacker. When an attacker sends the SYN packet, ServerIron ADX receives it and replies to it with SYN-ACK. If the attacker doesn't send an ACK to the ServerIron ADX, the handshake isn't completed with the ServerIron ADX. In this situation, the server never receives any packets from the attacking client and is oblivious to the attack.

If the SYN is from a valid client and not an attacker, ServerIron ADX completes the handshake and forwards the SYN to the host. ServerIron ADX creates a session at this time; only when the three-way handshake is complete.

NOTE
In software syn-proxy, throughput for syn-attack is 1.18Mbps per core.

Syn-Proxy auto control

Syn-Proxy can be explicitly enabled or disabled through a CLI command or setup to be automatically enabled when the TCP SYN packet arrival rate exceeds a configured threshold or disabled when the TCP SYN packet arrival rate falls below a configured threshold.

Difference between ServerIron ADX and JetCore Syn-Proxy Behavior

ServerIron ADX and JetCore-based ServerIron devices show different behavior with TCP Syn-Proxy.

A ServerIron ADX drops TCP SYN ACKs entering an interface where tcp syn-proxy is configured unless it can match those SYN ACKs to an existing session. The JetCore-based ServerIron devices forward them through. The behaviour of the ServerIron ADX provides enhanced protection against SYN attacks relative to the protection available from JetCore-based ServerIron devices.

If you want your ServerIron ADX to behave more like a JetCore-based ServerIron device, you can use any of the following three workarounds:

1. Enable syn-proxy on the server interface
2. Enable ip nat
3. Enable "server security-on-vip-only".

Configuring Syn-Proxy

This section contains the following sections:

NOTE
Syn-Proxy is not supported for IPv6 for releases earlier than 12.2.0.

NOTE
In a syn-proxy configuration for a local client, if an ARP entry for the client is not stored, the first TCP connection may need to retransmit none-syn packets since it may get dropped until the ServerIron ADX stores an ARP entry for the client. There will only be a performance impact for the very first connection.

NOTE
If you use **log** action inside access-list **deny** rules, then you cannot combine such an ACL with hardware-based syn-proxy on the same interface. To do so, you can either remove **log** action or disable hardware syn-proxy using the **server disable-hw-syn-cookie** command. Remember that if you disable hardware syn-proxy, you will harm syn-proxy performance.

NOTE
DSR is not supported with SYN-proxy and is supported with SYN-def.

Enabling SYN-Proxy

To activate Syn-Proxy, follow these steps:

1. Globally enable Syn-Proxy, using the following command:

   ```
   ServerIronADX(config)# ip tcp syn-proxy
   ```

   Syntax: ip tcp syn-proxy

   NOTE
   The **ip tcp syn-proxy** command must be executed at the global configuration level. If it is executed at the interface configuration level it will not take effect.

2. Configure a port and enter the interface configuration mode, using the following commands:

```
ServerIronADX(config)# interface ethernet 2/1
ServerIronADX(config-if-e1000-2/1)# ip tcp syn-proxy in
```

Syntax: interface ethernet <*slot number/port number*>

Syntax: ip tcp syn-proxy in

The **ip tcp syn-proxy** command can be configured for either a physical interface (as shown) or a ve interface.

Setting Attack-Rate-Threshold

A DoS attack threshold specifies the number of SYNs, without corresponding ACKs, the ServerIron ADX accepts before writing a warning message to the system log (every 60 seconds for the duration of the attack).

To configure a threshold value for the SYN attack rate, use the following command:

```
ServerIronADX(config)# server syn-cookie-attack-rate-threshold 2000
```

Syntax: server syn-cookie-attack-rate-threshold <*threshold value*>

<*threshold value*> is a number between 1 and 10,000,000. The default is 1000.

NOTE
If you do not configure a threshold, the ServerIron ADX does not write SYN attack messages to the log file.

If the number of SYNs (without corresponding ACKs), received on the ServerIron ADX exceeds the specified threshold of 2000, the ServerIron ADX writes the following message to the log file every 60 seconds for the duration of the attack:

```
"CRITICAL:  syn-cookie attack rate 2005 exceeds attack rate threshold 2000"
```

Setting SYN-Ack-Window-Size

To globally set the TCP window size that the ServerIron ADX uses on a SYN-ACK packet sent back to a client with SYN-Cookie, use the following command:

```
ServerIronADX(config)# server syn-proxy-syn-ack-window-size 5000
```

Syntax: [no] server syn-proxy-syn-ack-window-size <*value*>

<value> is the window size. The range from 1 to 65536. The default is 8192.

This command works with a syn-proxy configuration. By having syn-ack-window-size configured, the window size of the SYN-ACK packet sent from ServerIron ADX to client will have the configured <value>.

This feature can be used to prevent the client from sending HTTP-Gets before the server side 3-way handshake is established.

Setting Reset-Using-Client-MAC

NOTE
In this rare corner-case, the reset-using-client-mac command is needed to send a reset to the client.

To globally send a Reset to the client using the client MAC address on the interfaces where Syn-Proxy is enabled, use the following command:

```
ServerIronADX(config)#ip tcp syn-proxy reset-using-client-mac
```

Syntax: [no] ip tcp syn-proxy reset-using-client-mac

This command is useful only when the client cannot be reached using the ServerIron ADX default gateway and the default gateway of the server is different than the default gateway of the ServerIron ADX.

Retransmitting TCP SYNs

When Syn-Proxy is enabled, the ServerIron ADX completes the TCP three-way handshake with a connecting client prior to forwarding packets between the client and the destination server. This action allows the ServerIron ADX to forward to the server only packets associated with an established connection.

After completing the three-way handshake with the client, the ServerIron ADX sends a SYN to the destination server to attempt to establish a connection with the server. If the ServerIron ADX did not receive an ACK from the destination server within 8 seconds, the ServerIron ADX sent a TCP RESET to the client.

The ServerIron ADX performs retransmissions in 3-second intervals. If the ServerIron ADX does not receive an ACK from the destination server, it retransmits the SYN. After sending a SYN to the destination server, if the ServerIron ADX does not receive an ACK from the server after three seconds, the ServerIron ADX retransmits the SYN to the server. If the SYN is still unacknowledged after three more seconds, the ServerIron ADX retransmits the SYN to the server again. If after three retransmission attempts, the destination server still has not responded with an ACK, the ServerIron ADX sends a TCP RESET to the client to abort the connection.

Retransmitting the SYN to the server in this way allows the server to respond in case the initial SYNs sent by the ServerIron ADX are lost, without having to reset the connection with the client. The ServerIron ADX can retransmit SYNs for up to 65,536 pending connections concurrently.

This functionality is enabled by default when you enable syn-proxy. No CLI configuration is necessary. The output of show tcp-attack displays information about SYN retransmissions.

Setting the time range for a valid ACK packet

This feature sets a timer factor that determines the time range to accept a valid ACK packet. This feature is configured with the following command.

```
ServerIronADX(config)# ip tcp syn-proxy ack-validate-multiplier 3
```

Syntax: [no] ip tcp syn-proxy ack-validate-multiplier <*timer factor*>

The <*timer factor*> variable provides the contents of the timer factor in the following equation used to determine the time range used:

(timer factor +1) * 8 seconds

Example where the timer factor is set to 3.

The valid window is 3 + 1) * 8 = 32 seconds

Since we check the ACK packet using HASH data from two windows, the MAX time is 64 seconds.

Where the timer factor is set to 3, this HASH value will change every 32 seconds.

As a result, the valid ACK range = (timer factor +1) * 8 seconds * 2

Limiting syn-proxy feature to defined VIPs

With this feature enabled, the syn packets are dropped if a virtual server IP port is not defined under a VIP configuration. This feature is enabled with the following command.

```
ServerIronADX(config)# server syn-cookie-check-vport
```

Syntax: [no] server syn-cookie-check-vport

Setting the source MAC address

With this feature enabled, the SYN-ACK reply packets will have their source MAC address set to the MAC address of the ServerIron ADX. This can be helpful to avoid flooding in the case of a SYN to unknown uncast or broadcast address. This feature is enabled with the following command.

```
ServerIronADX(config)# server syn-cookie-set-sa
```

Syntax: [no] server syn-cookie-set-sa

Limiting the syn-proxy feature to VIP traffic only

This feature directs the ServerIron ADX to apply the Syn-Proxy feature to VIP traffic only (not to pass-through traffic). This feature is enabled with the following command.

```
ServerIronADX(config)# server security-on-vip-only
```

Syntax: [no] server security-on-vip-only

Dropping ACK packets with no data

This feature applies where Syn-Proxy is enabled. Configuring this feature causes ACK packets with no data to be dropped after the ServerIron ADX responds witha SYN-ACK to the client SYN. An ACK packet with data is forwarded to the BP and processed by the BP.

This feature is enabled with the following command.

```
ServerIronADX(config)# server virtual-name-or-ip www.altergo.com 207.95.55.1
ServerIronADX(config-vs-www.alterego.com)# port http drop-ack-with-no-data
```

Syntax: [no] port <tcp/udp-port > drop-ack-with-no-data

This feature is helpful in the event of a real SYN attack with a valid ACK packet sent but with no data packets afterwards

Setting a minimum MSS value for SYN-ACK packets

The default condition of the ServerIron ADX is to generate SYN-ACK packets with a Maximum Segment Size (MSS) that is equal or nearly equal to the client's MSS value. This process disregards the MSS value of the server. This can result in dropped packets or other unexpected behavior in situations where the MSS value of the server is smaller than the MSS value of the client.

This feature allows you to set the MSS value for SYN-ACK packets generated by the ServerIron ADX regardless of the client's MSS value. A minimum MSS value can be enabled in any of the following configurations:

- Global level – configures the TCP MSS value at the global level

- Virtual server lever – configures the TCP MSS value for all virtual ports under a specified virtual server
- Virtual port level – configures the TCP MSS value for a specified virtual port
- Destination IP – configures the TCP MSS value for pass-through traffic to a specified destination IP address

NOTE
tcp-mss will work when syn-proxy is enabled. If syn-proxy is turned off, tcp-mss will not take effect.

If the configured mininum MSS is larger than the client's actual MSS value, the ServerIron ADX will use the client's MSS value in SYN-ACK.

Hierarchy of operation

When multiple levels of the minimum MSS value are configured, the MSS value used by the ServerIron ADX is determined by the following hierarchy.

1. Virtual Port Level – Values configured at this level take precedence over any other MSS setting on the ServerIron ADX.
2. Virtual Server level – Only values configured at the Virtual Port level take precedence over MSS values configured at this level.
3. Global level – Values configured at this level take effect over all SYN-ACK packets generated by a ServerIron ADX unless the MSS value is configured at one of the levels previous described in 1, 2 or 3.

Setting the MSS value at the global level

To globally set the MSS value for all SYN-ACK packets generated by a ServerIron ADX, use the following command:

```
ServerIronADX(config)# tcp-mss 128
```

Syntax: [no] tcp-mss <*mss-value*>

The <*mss-value*> variable specifies MSS value for all SYN-ACK packets generated by the ServerIron ADX regardless of the client's MSS value. This value can be from 64 to 9216. Make sure that the IP MTU of the interfaces is always greater than the MSS value.

NOTE
When tcp-mss is configured at the global level, the same value will work for both IPv4 traffic and IPv6 traffic.

Setting the MSS value at the virtual server level

To set the MSS value for all of the ports under a virtual server on a ServerIron ADX, use the following command:

```
ServerIronADX(config)# server virtual-name-or-ip v1
ServerIronADX(config-vs-v1)# tcp-mss 128
```

Syntax: [no] tcp-mss <*mss-value*>

The <*mss-value*> variable specifies MSS value for all SYN-ACK packets generated by the ServerIron ADX for this virtual server regardless of the client's MSS value. This value can be from 64 to 9216. Make sure that the IP MTU of the interfaces is always greater than the MSS value.

Setting the MSS value at the virtual port level

To set the MSS value for a specific virtual port on a ServerIron ADX, use the following command:

```
ServerIronADX(config)# server virtual-name-or-ip v1
ServerIronADX(config-vs-v1)# port http tcp-mss 128
```

Syntax: [no] port <port-value> **tcp-mss** <*mss-value*>

The <*mss-value*> variable specifies MSS value for all SYN-ACK packets generated by the ServerIron ADX for the port specified by the <port-value> variable regardless of the client's MSS value. This value can be from 64 to 9216. Make sure that the IP MTU of the interfaces is always greater than the MSS value.

Setting the MSS value for pass-through traffic to a specified destination IP address

To set the MSS value for ServerIron ADX pass-through traffic to a specified destination IP address, use the following commands.

```
ServerIronADX(config)# tcp-mss 128 destination-ip 207.95.55.1
```

For IPv4

Syntax: [no] tcp-mss <*mss-value*> **destination-ip** <*ip-address*>

For IPv6

Syntax: [no] tcp-mss <*mss-value*> **destination-ipv6** <*ipv6-address*>

The <*mss-value*> variable specifies MSS value for all SYN-ACK packets that are ServerIron ADX pass-through traffic to a destination IP address specified by the <*ip-address*> variable. This value can be from 64 to 9216. Make sure that the IP MTU of the interfaces is always greater than the MSS value.

The <*ip-address*> or <*ipv6-address*> cannot be a Virtual server IP address.

Negotiated MSS value set

Once tcp-mss is configured with the minimum value, the ServerIron ADX will generate a negotiated MSS value in SYN-ACK base on the configured minimum MSS value. This MSS value will be the final MSS value after negotiation.

For example, if a user configures **tcp-mss 1200**, which is in the range of 1024 and 1440, a ServerIron ADX will use the lower 1024 as the negotiated MSS value in the SYN-ACK.

TABLE 9 MSS values for IPv4, IPv6 and IPv4 jumbo

	MSS value
IPv4	64, 256, 536, 966, 1024, 1440, 1452, 1460

TABLE 9 MSS values for IPv4, IPv6 and IPv4 jumbo

	MSS value
IPv6	64, 236, 516, 946, 1004, 1420, 1432, 1440
IPv4 Jumbo	256, 536, 966, 1024, 1452, 1460, 4038, 8960

Configuring Syn-Proxy auto control

Syn-proxy auto control operates the same as the normal Syn-proxy feature except that it is enabled and disabled based-on the arrival rate of TCP SYN packets on the ServerIron ADX. This is described in "Syn-Proxy auto control" on page 113. The following steps describe how to configure your ServerIron ADX for Syn-proxy auto control.

1. Set the SYN-Proxy auto control threshold levels – This procedure described in "Setting the SYN-Proxy auto control thresholds" on page 120, sets the thresholds for enabling and disabling Syn-Proxy during operation of the ServerIron ADX.
2. Set the interval time for counting TCP SYN packets – This procedure described in "Setting the interval time for counting TCP SYN packets" on page 121, sets the time period over which the thresholds set in Step 1 are evaluated.
3. Define Syn-Proxy on an in-bound interface – This is described in Step 2 of the procedure for "Enabling SYN-Proxy" on page 114.

Considerations for configuring Syn-proxy auto control

The following details concerning operation of the Syn-proxy feature should be considered when configuring the Syn-proxy auto control feature on a ServerIron ADX:

- All traffic including SLB and pass-through traffic is brought to a BP. Consequently, regardless of whether or not an interface has the **syn-proxy** feature enabled, if the threshold set for the rate of syns received per-second is exceeded for all ports on a ServerIron ADX, Syn-proxy auto control is enabled and will stay enabled as long as the rate remains above the configured off-threshold value.
- For interfaces that do not have the **syn-proxy** feature enabled, there will not be any syn attack protection even when Syn-proxy is enabled through auto control. Consequently, for the Syn-proxy auto control feature to work as expected, we recommend that **syn-proxy** be enabled on all interfaces.

Setting the SYN-Proxy auto control thresholds

To activate Syn-Proxy auto control, follow these steps:

Globally enable Syn-Proxy auto control by setting the thresholds for enabling and disabling Syn-Proxy as shown in the following command.

```
ServerIronADX(config)# ip tcp syn-proxy on-threshold 1000 off-threshold 500
```

Syntax: ip tcp syn-proxy on-threshold <on-threshold-value> off-threshold <off-threshold-value>

The **on-threshold** parameter is used to define the rate of syns received per-second (specified by the **<on-threshold-value>** variable) at which the Syn-Proxy feature is enabled on the ServerIron ADX.

The **on-threshold-value** variable is used with the **on-threshold** parameter and specifies the number of TCP SYN packets received per-second. When this value is exceeded for an interval time defined by the **server syn-attack-detection-interval** command, Syn Proxy is enabled on the ServerIron ADX. This value should be set to a much higher value than the normal TCP SYN packet arrival rate.

The **off-threshold** parameter is used to define the rate of syns per-second (specified by the **<off-threshold-value>** variable) at which the Syn-proxy feature is disabled (after being previously enabled) on the ServerIron ADX.

The **off-threshold-value** variable is used with the **off-threshold** parameter and specifies the number of TCP SYN packets received per-second. When the rate received drops below this value, the ServerIron ADX waits ten seconds and then disables Syn-proxy. The **<off-threshold-value >**variable must be less than the **<on-threshold-value>** variable.

Setting the interval time for counting TCP SYN packets

The rate at which Syn-proxy is enabled and disabled is determined by the thresholds set in the **ip tcp syn-proxy on-threshold <on-threshold-value> off-threshold <off-threshold-value>** command over the time period specified in the **server syn-attack-detection-interval** command. This interval is configured on the ServerIron ADX as shown in the following.

```
ServerIronADX(config)# server syn-attack-detection-interval 10
```

Syntax: server syn-attack-detection-interval <detection-interval-value>

The **<detection-interval-value>** variable defines the interval that is used to define the time for counting TCP SYN packets. The range of settings for this interval is 1 to 10 with each level representing 100 ms. Consequently, the interval can be from 100 ms to 1 second. If the interval value is smaller, the reaction time for enabling Syn-proxy is shorter, and the measurement of the TCP syn-packet arrival rate is less accurate. If the interval value is larger, the reaction time for enabling syn-proxy is longer, and the measurement of TCP syn-packet arrival rate is more accurate. The default interval value is 3 (in effect 300ms).

Displaying Syn-Proxy Commands

This section contains the following sections:

Displaying TCP Attack Information

The **show server tcp-attack** command displays attack information for connection rates counters.

```
ServerIronADX# show server tcp-attack
Connection counters:
Current conn rate =          0                    Max conn rate =          1

Attack counters:
Current attack rate =          0               Max attack rate =        0
Client-side counters:
         SYN rcvd =          6                      SYN-ACK sent =         6
     Valid ACKs rcvd =       3                   Invalid ACKs rcvd =         33
     Client pkt rcvd =      15                     Data pkt stored =         3
ACK without data dropp =       0
Destination-side counters:
            SYN sent =       3                      SYN-ACK rcvd =         3
  Duplicate SYN sent =       0              Duplicate SYN-ACK rcvd =         0
     Server pkt rcvd =      21                     Stored pkt sent =         0
```

Syntax: show server tcp-attack [debug | fast-path]

Displaying Server Traffic information

The **show server traffic** command displays four counters that help to analyze incoming traffic and determine the DOS attack occurrence. Be sure to issue show L4-traffic from the SSM CPU (not the MP).

```
ServerIronADX# show server traffic
Client->Server       = 3760614467  Server->Client       = 2169558899
Drops                =          0  Aged                =   17568293
Fw_drops             =          0  Rev_drops           =          0
FIN_or_RST           =  169210866  old-conn            =          0
Disable_drop         =          0  Exceed_drop         =          0
Stale_drop           =          9  Unsuccessful        =          0
SYN def/proxy RST    =          0  Server Resets       =          0
Out of Memory        =          0  Out of Memory       =          0
last conn rate       =          0  max conn rate       =      16283
last TCP attack rate =          0  max TCP attack rate  =          0
fast vport found     =          0  fast vport n found   =        477
Fwd to non-static FI =          0  Dup stale SYN        =          0

TCP forward FIN      =          0  TCP reverse FIN      =          0
Fast path FWD FIN    =          0  Fast path REV FIN    =          0
Fast path SLB SYN    =          0  Dup SYN after FIN    =          0
Duplicate SYN        =          0  Duplicate sessions   =          0
TCP ttl FIN recvd    =          0  TCP ttl reset recvd  =          0
Sessions in DEL_Q    =          0  Sess force deleted   =          0
Fwd sess not found   =          0  sess already in delQ =          0
Sess rmvd from delQ  =          0
New sess sync sent   =          0  New sess sync recvd  =          0
TCP SYN received     =          0  TCP SYN dropped      =          0
TCP SYN to MP        =          0  TCP SYN ACK to MP    =          0
TCP SYN ACK received =          0  TCP SYN ACK dropped  =          0
TCP pkt received     =          0  TCP pkt dropped      =          0
TCP pkt to MP        =          0

Dropped VIP pings    =          0
```

Syntax: show server traffic

TABLE 10 Field Descriptions for show L4-traffic

Field	Description
last conn rate	Rate of TCP traffic per second. This includes all TCP traffic, including TCP SYN DoS attacks
max conn rate	Peak rate of TCP traffic (per second) encountered on this device.
last TCP attack rate	Rate of TCP Dos attacks per second. This rate is delayed by 1 to 2 minutes.
max TCP attack rate	Peak rate of TCP DoS attacks (per second) encountered on this device. This rate is delayed by 1 to 2 minutes.

Displaying SYN Cookie Information

This **show server syn-cookie** command displays information about the SYN AKS that are sent and received.

```
ServerIronADX#show server syn-cookie
  CPU SYNs processed :          4         AXP SYNs processed :   92853225
   CPU SYN-ACKs sent :          4           AXP SYN-ACK sent :   92853225
 CPU Valid ACKs rcvd :   92951930        AXP Valid ACKS rcvd :  253002850
   Invalid ACKs rcvd :    1741885
          ACL passed :          0               ACL failed :          0
       Frags allowed :          0            Frags dropped :      73009
ACK without data dro :          0
       Invalid vport :          0
```

Syntax: show server syn-cookie

TABLE 11 Output Descriptions for show server syn-cookie

Field	Description
CPU SYNs rcvd	
AXP SYNs rcvd	Number of SYNs received on ServerIron ADX ports that have the Syn-Proxy feature enabled.
CPU SYN-ACKs sent	
AXP SYN-ACKs sent	Number of SYN ACKs sent from the ServerIron ADX to the client
CPU Valid ACKs rcvd	
AXP Valid ACKs rcvd	Number of valid ACKs received from the client.
Invalid ACKs rcvd	Number or invalid ACKs received from the client.
ACL passed	Number of ACL lookups that the ServerIron ADX passed.
ACL failed	Number of ACL lookups that the ServerIron ADX denied.
Frags allowed	Number of fragmented packets allowed.
Frags dropped	Number of fragmented packets dropped.
ACK without datp dro:	
Invalid vport	

DDoS protection

A Distributed Denial of Service (DDoS) attack is employed to cause a denial of service to legitimate users by consuming all or most of the CPU and memory resources on a ServerIron ADX or on real servers. The ServerIron ADX provides protection and prevents well-known DDoS attacks such as Xmas-tree attack, syn fragment, address sweep and others. The ServerIron ADX prevents these attacks by defining filters for each type of attack coupled with a **drop** or **log** action. These filters are then bound to an interface. All packets that match the filter on the bound interface are dropped or logged as defined in the configuration. Filters can be defined according to a generic rule as shown in "Configuring a Generic Rule" on page 125 or applied from built-in rules as described in Table 13, Table 15, Table 16 and Table 17. Filters are applied to IPv4 and IPv6 traffic where appropriate.

The following sections describe how to configure a security filter, define rules within a security filter and bind a security filter to an interface.

Configuring a security filter

Configuring a a security filter requires you to define it by name and configure rules within it as shown in the following.

```
ServerIronADX(config)# security filter filter1
ServerIronADX(config-sec-filter1)#rule xmas-tree drop
```

Syntax: security filter <*filter-name*>

The <*filter-name*> variable specifies the filter being defined that will then be bound to a port.

The **rule** command defines the attack method that is being filtered for. For each rule, you can configure whatever action needs to be taken if an attack occurs. The ServerIron ADX can log the attack and drop the attacking packet. Rules that can be used are described in Tables 12 thorugh 17 of this chapter.

Some rules are hardware-based and are programmed in the CAM. For these rules, the default action is to drop the packet. These rules cannot be logged, and no message can be logged when an attack occurs. But there are counters that you can check to determine if an attack has occurred.

Example

```
ServerIronADX(config)# security filter filter1
ServerIronADX(config-sec-filter1)# rule xmas-tree log
ServerIronADX(config-sec-filter1)# rule address-sweep 1 3 drop log
```

NOTE
There is no set limit on the number of filters that can be configured on a ServerIron ADX but a maximum of 10 rules can be bound to a single interface. The global limit depends upon the available memory.

Configuring a Generic Rule

Apart from regular rules, such as those configured above, there is also a *generic* rule. A generic rule needs to be defined before it can be bound to a filter. In the following example, a rule (**gen1**) is configured to match a tcp packet with source-ip greater than 10.10.1.101, a tcp dest-port greater than 20 and a string "400" at the 3rd byte offset from l4-data.

```
ServerIronADX(config)# security generic gen1
ServerIronADX(config-sec-gen-gen1)# ip-source gteq ip 10.10.1.101
ServerIronADX(config-sec-gen-gen1)# tcp-dest gt val 20
ServerIronADX(config-sec-gen-gen1)# l4-data 3 eq str "400"
```

Syntax: {no} security generic <*generic-rule-name*>

The <*generic-rule-name*> variable specifies the generic rule defined that will then be bound to a filter.

The following conditions can be applied to any of the fields in the mac-header, ip-header, l4-header (TCP/UDP), and l4-data offset to create generic rules:

eq equals

gt greater-than

gteq greater-than-or-equals

lt less-than

lteq less-than-or-equals

neq not-equals

The configured generic rule will have to be bound to a filter, to take effect.

```
ServerIronADX(config)# security filter filter1
ServerIronADX(config-sec-filter1)# rule generic gen1 drop
```

Syntax: {no} rule generic <*generic-rule-name*> **[log | no-log] [drop | no-drop]**

The *<generic-rule-name>* variable is the name of the preciously defined generic rule that you want to bind to a filter:

The **log** parameter directs the ServerIron ADX to log traffic on the bound interface that matches the generic rule specified by the configured *<generic-rule-name>*. The **no-log** parameter disables this function.

The **drop** parameter directs the ServerIron ADX to drop traffic on the bound interface that matches the generic rule specified by the configured *<generic-rule-name>*. The **no-drop** parameter disables this function.

Table 13 describes some attack types that require a generic rule.

TABLE 12 Common attack types that require a generic rule

Attack Type	Description
Information tunneling	Attacker attempts to pass information in and out of the network incognito. Packets appear to be performing one function. In reality, they are performing another function. For example, a remote user may be engaged in a root shell session on a protected host, but all transmissions appear to be ICMP echo requests and replies. Use security generic to handle this attack type.
Well Known Attacks	There are many documented attacks that can be identified by using a pattern, also known as a signature. Use security generic for this attack type. It provides you the flexibility of locating attacks having a pattern.

Configuring a rule for common attack types

As described in "Configuring a Generic Rule" on page 125, you can create a custom rule to manage DDoS attacks. In addition, ServerIron ADX has built-in rules to manage common attack types. In this case, the **rule** command is used with a <rule-name> variable specified in Table 13.

The following example configures a the "filter1" security filter with a rule to drop packets that are associated with a "xmas tree" attack.

```
ServerIronADX(config)# security filter filter1
ServerIronADX(config-sec-filter1)#rule xmas-tree drop
```

Syntax: [no] rule <*rule-name*> **[log | no-log] [drop | no-drop]**

The *<rule-name>* variable is specified as one of the options described in Table 13.

The **log** parameter directs the ServerIron ADX to log traffic on the bound interface that matches the rule specified by the configured *<rule-name>*. The **no-log** parameter disables this function.

The **drop** parameter directs the ServerIron ADX to drop traffic on the bound interface that matches the rule specified by the configured *<rule-name>*. The **no-drop** parameter disables this function.

TABLE 13 Rules for common attack types and descriptions

Attack Type	Description
syn-fragments	A SYN-fragment attack floods the target host with SYN-packet fragments. The host stores the fragments, in order to reassemble them. By not completing the connection, and flooding the server or host with such fragmented SYN packets, the host's memory buffer would eventually fill up. A SYN packet need not be fragmented. It is very small. Use **syn-fragments** to drop any SYN packet with the more-fragments bit on.
syn-and-fin-set	Attacker sends a packet with both SYN and FIN bits set to see what kind of system reply is returned, and then use the system information for further attacks using known system vulnerabilities. Use **syn-and-fin-set** to drop packets having both the SYN and FIN bits set in the flags field
tcp-no-flags	TCP packets are normally sent with at least one bit set in the flags field. Use **tcp-no-flags** to drop TCP packet with a missing or malformed flags field.
icmp-fragments	ICMP fragments can be used to exploit the vulnerabilities in the packet reassembly code. Enabling **icmp-fragments** allows you to create a rule that can be configured to drop or log fragmented ICMP packets.
deny-all-fragments	IP fragments can be used to exploit the vulnerabilities in the packet reassembly code of specific IP stack implementations. When you enable **deny-all-fragments**, the ServerIron ADX drops all IP fragments.
land-attack	Attacker sends spoofed SYN packets containing the destination IP as the source-IP address. Flooding a system with such empty connections can overwhelm the system, causing DoS. Use land-attack to drop packets containing the same IP address as both the source and destination IP addresses.
ping-of-death	If the sum of "Fragment Offset" and "Total length" fields in the IP header of each IP fragment is larger than 65,535, then the packet is invalid, and such packet will be dropped.

TABLE 13 Rules for common attack types and descriptions

fin-with-no-ack	TCP packets with a FIN flag normally have an ACK bit set. Use **fin-with-no-ack** to drop TCP packet where FIN flag is set, but the ACK bit is not set.
large icmp	ICMP packets greater than 1500 bytes.
unknown-ip-protocol <*value*>	Protocol 101 and above are currently reserved and undefined. Attackers sometimes use protocol values that are not valid protocols. Use **unknown-ip-protocol** <value> to drop packets where protocol field is set to 101 or greater. Replace the <value> with the actual protocol number. If the ServerIron ADX gets an IP protocol packet matching that number, the packet will be dropped.
address-sweep <*dest-ip*> <*hold-down-interval*>	Attacker scans the network for information behind the ServerIron ADX, uncovering an address to target. For example, sending ping requests to 10.1.1.1 through 100. A reply from any device indicates a server exists. Use address-sweep <dest-ip> <hold-down-interval> to log the number of different addresses being accessed from one remote source. If the same client sends a request to different IP addresses, then the ServerIron ADX keeps track of the number of IP address the client is trying to access. If the number exceeds the configured <dest-ip> limit within a 5-second timer, then all the packets from that client will start being dropped (reset) for the specified <hold-down-interval>. The timer is internal to the ServerIron ADX, and it does not start when the client accesses the first IP address. For example, consider the command address-sweep 2 8. A client can access only a maximum of two IP addresses. As soon as it accesses the third IP within a 5-second interval, the ServerIron ADX will hold the client down for 8 minutes. Accessing two IP addresses within 5 seconds is permissible in this example. Use security **net-scan-mon-interval** <*seconds*> to change the timer default (5 seconds).
port-scan <*dest-ip-or-port-pair*> <*hold-down-interval*>	Attacker sends traffic using the same source IP to different ports on the same destination IP, with the intent of identifying a service to target. The ServerIron ADX tracks the number of IP destination and port pairs. For example, if a client accessed IP 1 and port 0, that counts as one <dest-ip-or-port-pair>. Use port-scan to internally log the number of different ports scanned from one source. If the number exceeds a configured <dest-ip-or-port-pair> value (for example, 10 ports in 5 seconds), the ServerIron ADX will flag it as an attack and reject all traffic from that source for the <hold-down-interval>. The default internal timer value is 5 seconds. Use **security net-scan-mon-interval** <seconds> to change the timer default.
xmas-tree	A Xmas tree attack is detected when a packet with the URG,PSH & FIN flags set is detected.
icmp-type	Different types and subtypes of ICMP can be used to attack or to gain knowledge about the host or network, which would then be used for an attack. For example, ICMP timestamp (type 13) will elicit a timestamp reply from Unix systems, but Microsoft Windows would not do so. A hacker can then attack known vulnerabilities of the system. Use icmp-type to configure an ICMP software rule to drop specific ICMP types and subtypes.

Configuring a rule for ip-option attack types

ServerIron ADX has a set of built-in rules to manage ip-option attack types. In this case, the **rule** command is used with a <ip-option-attack> variable specified in Table 14.

The following example configures the "filter2" security filter with a rule to drop packets that are associated with a **ip-option record-route** attack.

```
ServerIronADX(config)# security filter filter2
ServerIronADX(config-sec-filter2)#rule ip-option record-route drop
```

Syntax: [no] rule ip-option *<ip-option-attack>* **[log | no-log] [drop | no-drop]**

The *<ip-option-attack>* variable is specified as one of the options described in Table 14.

The **log** parameter directs the ServerIron ADX to log traffic on the bound interface that matches the rule specified by the configured *<ip-option-attack>*. The **no-log** parameter disables this function.

The **drop** parameter directs the ServerIron ADX to drop traffic on the bound interface that matches the rule specified by the configured *<ip-option-attack>*. The **no-drop** parameter disables this function

TABLE 14 ip-option attack types and descriptions

Attack Type	Description
ip-option record-route	The **record-route** option records the path of the packet, which an attacker can analyze to learn details about a network's addressing scheme and topology. Use **ip-option record-route** to drop packets with IP option 7 (record route) set.
ip-option strict-source-route	The **strict-source** option provides a means for the source of a packet to supply routing information to the gateways forwarding the packet to the destination, and to record the route information. With this option, an attacker can gain knowledge on the network's addressing scheme. Use **ip-option strict-source-route** to drop packets having IP option 9 (strict source routing).
ip-option loose-source-route	The **loose-source** option provides a means for the source of the packet to supply routing information to be used by the gateways in forwarding the packet to the destination. This option is different from strict-source route because gateway or host IP is allowed to use any route of any number of other intermediate gateways to reach the next address in the route. With this option, an attacker can gain knowledge on the network's addressing scheme. Use **ip-option loose-source-route** to drop packets that have IP option 3 (loose source routing).
ip-option timestamp	Use ip-option timestamp to drop packets where IP option list includes option 4 (Internet timestamp).
ip-option stream-id	The stream-ID option provides a way for the 16-bit SATNET stream identifier to be carried through networks that do not support the stream concept. Use **ip-option stream-id** to drop packets where the IP option is 8 (stream ID).

Configuring a rule for icmp-type options

ServerIron ADX has a set of built-in rules to manage icmp-type options. In this case, the **rule-icmp-type** command is used with a <*icmp-option-attack*> variable specified in Table 15.

The following example configures the "filter3" security filter with a rule to drop packets that contain the **icmp-type echo-reply** type.

```
ServerIronADX(config)# security filter filter3
ServerIronADX(config-sec-filter3)# rule icmp-type echo-reply drop
```

Syntax: [no] **rule icmp-type** <*icmp-type*> **[log | no-log] [drop | no-drop]**

The <*icmp-type*> variable can be one of the options described in Table 15

The **log** parameter directs the ServerIron ADX to drop traffic on the bound interface that matches the rule specified by the configured <*icmp-type*>. The **no-log** parameter disables this function.

The **drop** parameter directs the ServerIron ADX to drop traffic on the bound interface that matches the rule specified by the configured <*icmp-type*>. The **no-drop** parameter disables this function

TABLE 15 icmp option types and descriptions

ICMP Option Type	Description
icmp-type addr-mask	icmp type 17: addr-mask timestamp-reply icmp type 14: timestamp-reply
icmp-type addr-mask-reply	addr-mask-reply: icmp type 18: addr-mask-reply
icmp-type destination <type> i	icmp type 3: destination-unreachable. The <type> variable is specified as one of the following values. **admin-prohibited** code 13: admin-prohibit **fragment-needed** code 4: fragment-needed **host-admin-prohibited** code 10: destination-host-admin-prohibited **host-precedence-violation** code 14: host-precedence-violation **host-unknown** code 7: destination-host-unknown **host-unreachable** code 1: host-unreachable **host-unreachable-for-tos** code 12: host-unreachable-for-tos **net-admin-prohibited** code 9: destination-network-admin-prohibited **net-unknown** code 6: net-unknown **net-unreachable** code 0: network-unreachable **network-unreachable-for-tos** code 11: network-unreachable-for-tos **port-unreachable** code 3: port-unreachable **precedence-cutoff** code 15: precedence-cutoff-in-effect **protocol-unreachable** code 2: protocol-unreachable **route-fail** code 5: route-fail **source-host-isolated** code 8: source-host-isolate
icmp-type echo-reply	icmp type 0: echo-reply
icmp-type echo-request	icmp type 8: echo-request
icmp-type info-reply	icmp type 16: information-reply
icmp-type info-request	icmp type 15: information-request
icmp-type param-problem	icmp type 12: parameter-problem
icmp-type redirect	icmp type 5: redirect

TABLE 15 icmp option types and descriptions

icmp-type router-advertisement	icmp type 9: router-advertisement
icmp-type r outer-selection	icmp type 10: router-selection
icmp-type source-quench	icmp type 4: source-quench
icmp-type time	icmp type 11: time-exceeded
icmp-type timestamp	icmp type 13: timestamp
icmp-type timestamp-reply	icmp type 14: timestamp-reply

Configuring a rule for IPv6 ICMP types

ServerIron ADX has a set of built-in rules to manage IPv6 icmp types. In this case, the **rule** command is used with a <*icmp-option*> variable specified in Table 16.

The following example configures the "filter4" security filter with a rule to drop packets that acontain the **icmpv6-option** type **echo-reply**.

```
ServerIronADX(config)# security filter filter4
ServerIronADX(config-sec-filter4)#rule icmp-type echo-reply drop
```

Syntax: [no] rule ip-option <*icmpv6-type*> **[log | no-log] [drop | no-drop]**

The *<ipv6-type>* variable is specified as one of the options described in Table 16.

The **log** parameter directs the ServerIron ADX to drop traffic on the bound interface that matches the rule specified by the configured *<icmpv6-type>*. The **no-log** parameter disables this function.

The **drop** parameter directs the ServerIron ADX to drop traffic on the bound interface that matches the rule specified by the configured *<icmpv6-type>*. The **no-drop** parameter disables this function

TABLE 16 ICMPv6 types and descriptions

Attack Type	Description
cpa	ICMP type 149: Certification Path Advertisement.
cps	ICMP type 148: Certification Path Solicitation
echo-reply	ICMP type 129: echo-reply
echo-request	ICMP type 148: echo-request
mra	ICMP type 151: Multicast Router Advertisement
mrs	ICMP type 152: Multicast Router Solicitation
mrt	ICMP type 153: Multicast Router Termination
neighbor-advertisement	ICMP type 136: neighbor-advertisement
neighbor-solicitation	ICMP type 135: neighbor-solicitation
private	ICMP type 200: Private experimentation
private1	ICMP type 201: Private experimentation
redirect-message	ICMP type 137: redirect-message

TABLE 16 ICMPv6 types and descriptions

reserved	ICMP type 255: reserved for expansion
router-advertisement	ICMP type 134: router-advertisement
router-solicitation	ICMP type 133: router-solicitation

Configuring a rule for IPv6 ext header types

ServerIron ADX has a set of built-in rules to manage IPv6 header types. In this case, the **rule** command is used with a <*ipv6-ext-header-type* > variable specified in Table 17.

The following example configures the "filter5" security filter with a rule to drop packets that contain the **ipv6-ext-header** type **esp**.

```
ServerIronADX(config)# security filter filter5
ServerIronADX(config-sec-filter5)#rule ipv6-ext-header-type esp drop
```

Syntax: [no] rule ipv6-ext-header-type <*ipv6-ext-header-type*> **[log | no-log] [drop | no-drop]**

The *<ipv6-ext-header-type>* variable is specified as one of the options described in Table 17.

The **log** parameter directs the ServerIron ADX to drop traffic on the bound interface that matches the rule specified by the configured *<ipv6-ext-header-type>*. The **no-log** parameter disables this function.

The **drop** parameter directs the ServerIron ADX to drop traffic on the bound interface that matches the rule specified by the configured *<ipv6-ext-header-type>*. The **no-drop** parameter disables this function

TABLE 17 IPv6 ext header types and descriptions

Attack Type	Description
ah	Authentication Header Option
cfg-hdr0-num	Configurable extension header code 0
cfg-hdr1-num	Configurable extension header code 1
cfg-hdr2-num	Configurable extension header code 2
cfg-hdr3-num	Configurable extension header code 3
destination-option	Destination Options (with Routing Options)
esp	Encapsulation Security Payload Header
hop-by-hop	Hop-by-Hop option
mobility-header	Mobility Header option
no-next-header	No Next Header
routing-header	Routing Header option
unknown-header	Unknown headers are those that are not listed in the above header types and TCP/UDP/ICMPv6.

Binding the filter to an interface

To implement a filter, it must be bound to an interface. It will then be applied globally to all interfaces on the ServerIron ADX. To bind a filter to an interface, use the following command:

```
ServerIronADX(config-if-e1000-1/2)# security apply-filter filter1
```

Syntax: **security apply-filter** *<filter-name>*

The *<filter-name>* variable specifies filter that you want to apply on the ServerIron ADX. A maximum or 10 filters can be bound to a single interface.

Clearing DOS attack statistics

Use **clear statistics dos-attack** to reset counters for ICMP and TCP SYN packet burst thresholds.

Syntax: **clear statistics dos-attack**

Clearing all DDOS Filter & Attack Counters

Use **security clear all-dos-filter-counters** to reset all DDOS Filter and Attack Counters.

Syntax: **security clear all-dos-filter-counters**

Logging for DoS attacks

Use the **show log** command to display the logging information and notice the attack type hits:

For each log event taking place for software rules, the ServerIron ADX sends a syslog message and an SNMP trap. The system logs every 1 second time period, but only the difference is logged (not cumulative totals). For example, assume 5 packets are dropped within 1 second. The system logs 5. Then, 2 packets are dropped during the next second. The system logs 2 (not 7).

Use **show security hold**:

Use **show security net-scan-sessions**:

```
BP # show sec net-scan-sessions <number to be skipped>
IP address                          Attack Type     Number Scanned
10.10.1.101->10.10.1.151             port-scan            1
```

The number scanned indicate the number of ports client 10.10.1.101 has accessed on IP 10.10.1.151 (which is the VIP in the example).

Similarly for address-sweep:

```
BP #show sec net-scan-sessions 0
IP address                          Attack Type     Number Scanned
10.10.1.101                         address-sweep        2
```

The above example tells you that client 10.10.1.101 has accessed 2 destination IPs in the past 1 monitoring interval.

Displaying security filter statistics

You can display security filter statistics as shown.

```
ServerIronADX# show security filter-statistics
Filter             |Type           |Log Cnt   |Drop Cnt
dos-filter       |icmp-type   |0             |0

Cumulative Statistics
attack-type = log-count, drop-count
ip-options          = 0, 0
icmp-type           = 0, 0
address-sweep       = 0, 0
port-scan           = 0, 0
generic             = 0, 0
filter-dns          = 0, 0

Attack-type = Attack-count
ipv6-ext-header    = 1201
icmpv6-type-All    =  321
icmpv6-type-NS     =  221
icmpv6-type -NA    =   60
icmpv6-type-RS     =   24
icmpv6-type-RA     =   16
large-icmp         =    0
unknown-ip-proto   =    0
xmas-tree          =    0
tcp-no-flags       =    0
syn-fragments      =    0
syn-and-fin-set    =    0
deny-all-fragments =    0
fin-with-no-ack    =    0
icmp-fragment      =    0
land-attack        =    0
ping-of-death      =    0
```

The counters shown for the **show security filter-statistics** command display the DDoS attack types and the number of packets that have been counted, logged or dropped for each type.

Syntax: show security filter-statistics

Address-sweep and port-scan logging

The ServerIron ADX provides a log message for address-sweep and port-scan. When the ServerIron ADX detects either of these attacks, the SSM CPU will send a message to the MP indicating the particular IP will be held down for the specified time interval.

Log example:

```
Security: Address-sweep attack detected!Holdown 10.10.1.101 for 2 min
```

Chapter

Secure Socket Layer (SSL) Acceleration 6

ServerIron ADX supports integrated hardware-based SSL acceleration. This chapter describes how to configure a ServerIron ADX for SSL acceleration in SSL Termination or SSL Proxy mode.

SSL support on the ServerIron ADX includes support for SSLv2, SSLv3, and TLS1.0.

SSL overview

The Secure Sockets Layer (SSL) protocol was developed by Netscape to provide security and privacy between client and server over the Internet. SSL supports server and client certificate verification, allowing protocols such as HTTP, FTP, and Telnet to be run on top of the verification process. SSL negotiates encryption keys and authenticates the server before data is exchanged by higher-level applications.

The SSL "handshake" is a key concept in this protocol. The handshake consists of two phases: server authentication, and an optional client certificate verification. In server authentication, the server sends its certificate and cipher preferences to a client that has made a request. The client then generates a master key, encrypts it with the public key of the server, and returns the encrypted master key to the server.

The server recovers the master key and authenticates itself to the client by returning a message encrypted with the master key. Subsequent data is encrypted and authenticated with keys derived from this master key. In the client certificate verification phase (which is optional), the server sends a challenge to the client, who authenticates itself to the server by returning a digital signature with its public-key certificate.

A variety of cryptographic algorithms are supported by SSL. During the "handshaking" process, the DSA public-key cryptosystem is used. After the exchange of keys, a number of ciphers are used that include RC4 and triple-DES for data encryption, and the SHA-1 and MD5 digest algorithm for message authentication.

Public Key Infrastructure (PKI)

In cryptography, a public key infrastructure (PKI) is an arrangement that provides for trusted third party vetting of, and vouching for, user identities. It also allows binding of public keys to users. This is usually carried out by software at a central location, together with other coordinated software at distributed locations. The public keys are typically in certificates.

The term PKI may mean both the certificate authority and related arrangements as well as, more broadly (which can sometimes be confusing), the use of public key algorithms in electronic communications. The latter meaning is erroneous since PKI methods are not required to use public key algorithms.

Asymmetric cryptography

This method alters information so that the key used for encryption is different from the key used for decryption. Encrypted information is unintelligible to unauthorized parties.

Certificate Authority (CA)

The certificate authority (CA) issues and manages security credentials and public keys for message encryption within a network.

Certificate Revocation List (CRL)

The CRL is a list of subscribers paired with their digital signature status, specifically any revoked certificates and the reason for the revocation.

Cipher suite

A cipher specifies the suite of cryptographic algorithms to be used for key exchange, bulk encryption, and message authentication.

Digital certificate

A digital certificate is a digital document that is generally stored and administered in a central directory. It contains the certificate holder's name, a serial number, expiration dates, public key, and the digital signature of the certificate issuing authority.

Digital signature

A digital (electronic) signature authenticates the identity of the sender, ensures that the original content of the message is unchanged, is easily transportable, cannot be easily repudiated, cannot be imitated, and can be automatically time stamped.

Key

When used in the context of cryptography, a key is a series of random numbers used by a cryptographic algorithm to transform plaintext data into encrypted data, and vice versa.

Key pair

A pair of digital keys - one public and one private - used for encrypting and signing digital information.

Private key

A cryptographic key known only to the user, employed in public key cryptography in decrypting or signing information. One half of a key pair.

Public key

The other half of a key pair, a public key is held in a digital certificate. Public keys are usually published in a directory. Any public key can encrypt information; however, data encrypted with a specific public key can only be decrypted by the corresponding private key.

NOTE
We recommend that you always back up your SSL certificate keys. These keys may be lost in the event of module failure.

SSL acceleration on the ServerIron ADX

The ServerIronADX SSL module provides hardware-accelerated encryption and decryption services to clients. The ServerIronADX sits between clients and servers and all client traffic is terminated on the switch. When traffic is decrypted, the ServerIronADX analyzes the data and selects a server where the connection traffic can be forwarded. The ServerIronADX then opens a new connection to the server and passes all data to this server. On the return path, the ServerIronADX receives all data from the server, encrypts it, and forwards it to the client. For every incoming connection from the client, the ServerIronADX maintains an additional connection to the server. Both connections are completely separate. The ServerIron ADX essentially acts as a proxy.

SSL acceleration on the ServerIron ADX can be configured to operate in either of the following two modes:

- **SSL Termination Mode** – In SSL Termination mode, an SSL connection is maintained between a client and a ServerIron ADX. The connection between the ServerIron ADX and the server is not encrypted.
- **SSL Full Proxy Mode** – In SSL Full Proxy mode, one SSL connection is maintained between a client and a ServerIron ADX and a separate SSL connection is maintained between a ServerIron ADX and a server. This connection allows for traffic encryption to be maintained all the way from the client to the server and back.

For details on how to configure a ServerIronADX for SSL Termination and Proxy modes, see Configuring Real and Virtual Servers for SSL Termination and Proxy Mode 172 and for examples of how to create the configurations shown in this section see Configuration Examples for SSL Termination and Proxy Modes 176.

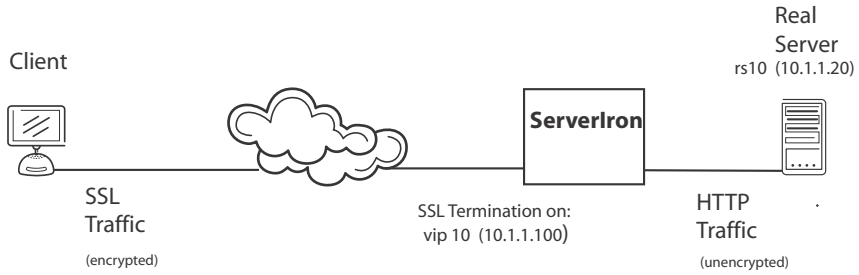
SSL Termination Mode

In this mode, the ServerIron ADX terminates the SSL connections, decrypts the data, and sends clear text to the server. The ServerIron ADX offloads the encryption and decryption services from the server CPU and performs them in hardware, thereby offloading the burden from the server.

The ServerIronADX maintains an encrypted data-channel with the client and a clear-text data channel with the server.

Figure shows a topology that terminates SSL on the ServerIron ADX.

FIGURE 9 ServerIron ADX SSL Termination



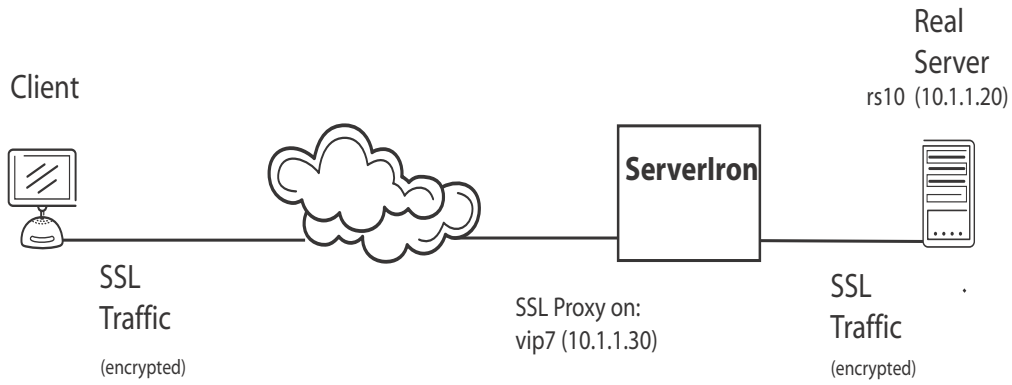
SSL Proxy Mode

In full SSL proxy mode, a ServerIronADX maintains encrypted data channels with the client and server. The ServerIronADX maintains an SSL session with the client and a separate one with the server. This maintains total SSL security between client and server.

This is useful in a configuration where you want to maintain full SSL security between a client and server and also have the ServerIronADX perform L7 processing and security to application traffic. This works because after the SSL connection is terminated at the ServerIronADX and before it enters the SSL connection with the server, it is in clear-text within the ServerIronADX where it can be subject to L7 inspection.

Figure 10 shows the basic topology for a configuration of the full SSL proxy mode.

FIGURE 10 ServerIron ADX SSL Proxy



ServerIron ADX SSL

This section describes the SSL features used in configuration of a ServerIron ADX for SSL acceleration.

ServerIron ADX keypair file

The keypair file specifies the location for retrieving the SSL asymmetric key pair, during an SSL handshake. You can create a keypair file by generating a key pair locally on the ServerIron ADX or import a pre-existing key pair, using secure copy (SCP).The key pair is stored in the flash memory and is not deleted during a power cycle. The ServerIron ADX supports keys in PEM (Privacy Enhanced Mail) or PKCS12 (Public Key Cryptography Standard 12) formats.

NOTE
ServerIronADX supports keys in PEM (Privacy Enhanced Mail) or PKCS12 (Public Key Cryptography Standard 12) formats.

Digital certificate

A digital (electronic) signature authenticates the identity of the sender, ensures that the original content of the message is unchanged, is easily transportable, cannot be easily repudiated, cannot be imitated, and can be automatically time stamped. The certificate contains the public part of the RSA key, which must be the same as the key in the keypair file. The public key used in the certificate file must match the key pair that is associated with the profile.

If the public key of the certificate does not match the key pair defined under the profile, the ServerIron ADX does not accept the connection.

If the public key in the certificate file does not match the key in the keypair file, a ServerIron ADX does not accept the command and displays the following error message:

```
"Error reading certificate from file <certfile-name>"
```

The certificate file specifies the location for retrieving the digital certificate the ServerIron ADX presents to the client for every new SSL handshake request. The certificate is stored in the flash memory, and is not deleted upon a power cycle.

You can create a certificate file locally or you can import it. You can also generate a Certificate Signing Request (CSR) and have it signed by a known CA to create a certificate and then import it. See the section "certificate management" for an overview of each process.

Chained certificates

In an ideal world, a Certificate Authority signs and issues a certificate directly to a server. The server loads this certificate and whenever a client makes a connection to it, this certificate is presented. The client already has a copy of the CA's public certificate and verifies the server certificate against it.

For manageability and security reasons, CAs may not sign server certificates directly. Many times, the root CA signs an intermediate CA that in turn signs the server certificates. When this happens, a certificate chain is created. There can be multiple levels of intermediate CA certificates.

Three level chain

- CA ----> 1st level Intermediate CA ----> server certificate

Four level chain

- CA ---> 1st level Intermediate CA ---> 2nd level Intermediate CA ---> server certificate

The end clients, including Mozilla, Firefox and Internet Explorer, always have a copy of the well-known parent CA's certificate. They, however, may not have the intermediate CA's certificates.

Configuring SSL on a ServerIron ADX

When configuring a ServerIron ADX for either SSL Termination mode or SSL Proxy mode, you must perform each of the following configuration tasks:

- Obtain a Keypair File – This section describes how to obtain an SSL asymmetic key pair. You can generate an RSA key pair or import an existing key pair. See "Obtaining a ServerIron ADX keypair file" on page 140.
- Certificate Management – This section describes various methods for obtaining a digital certificate and the methods for importing Keys and Certificates. See "Certificate management" on page 141.
- Basic SSL Profile Configuration – This section describes how to perform the minimum SSL profile configuration. See "Basic SSL profile configuration" on page 164.
- Advanced SSL Profile Configuration – This section describes additional SSL profile configuration parameters. See "Advanced SSL profile configuration" on page 166.
- Configure Real and Virtual Servers for SSL Termination and Proxy Mode – This section describes the configuration details required to configure the Real and Virtual servers for SSL on a ServerIron ADX. See "Configuring Real and Virtual Servers for SSL Termination and Proxy Mode" on page 172.
- Configuring Other Protocols with SSL– This section describes how to configure other popular protocols such as LDAPS, POP3S and IMAPS with SSL acceleration. See "Other protocols supported for SSL" on page 184
- Configure System Max Values – This section describes how to configure system max values for SSLv2 connection rate and memory limit for SSL hardware buffers. See "Configuring the system max values" on page 185.

Obtaining a ServerIron ADX keypair file

The keypair file specifies the location for retrieving the SSL asymmetric key pair, during an SSL handshake. You can either generate an RSA keypair file on a ServerIron ADX or import a pre-existing key pair, using secure copy (SCP).The key pair is stored in the flash memory and is not deleted during a power cycle.

To generate an RSA keypair file, enter the following command.

```
ServerIronADX# ssl genrsa rsakey-file 1024 mypassword
```

Syntax: ssl genrsa *<file-name> <key-strength> <password>*

The <file-name> variable specifies the name of the keypair file. The file name can be up to 24 characters in length. The file name supports special characters like '-',' _', '$', ", '%', '&', and '!'. It does not support spaces and '/' characters.

The <key-strength> variable specifies the Key strength (number of bits) for the RSA key pair. The RSA key strength should be 512, 768, 1024 or 2048.

NOTE
The ServerIron ADX does not support key strength greater than 2048 bits.

The <password> variable specifies the password to the file. The length of password should not exceed 64 characters.

Once a key pair is generated it can be saved for backup on your server by exporting it as described in "Importing keys and certificates" on page 148.

Also, you can import a keypair file (instead of generating it) as described in "Importing keys and certificates" on page 148.

NOTE
The ServerIron ADX supports keys in PEM (Privacy Enhanced Mail) or PKCS12 (Public Key Cryptography Standard 12) formats.

Certificate management

All configuration options used with the SSL acceleration features of the ServerIron ADX require that you obtain a Certificate and upload it to the system. The following methods can be used to obtain as certificate.

- "Generating a Self-Signed Certificate" on page 141
- "Using CA-signed certificates" on page 142
- "Exporting Web Server Certificates" on page 143

Once a digital certificate and a keypair are obtained you can Import them to the ServerIron ADX using the procedures described in "Importing keys and certificates" on page 148. This section also describes how to configure a list of certificates that have been revoked by a CA in "Importing keys and certificates" on page 148.

Generating a Self-Signed Certificate

Before generating a self-signed certificate, you must obtain an RSA key pair as described in Obtaining a ServerIron ADX keypair file 140

Once you've obtained the RSA key pair, you can generate a self-signed certificate as shown in the following example.

```
ServerIronADX# ssl gencert certkey testkey signkey testkey brocade123 testcert
You are about to be asked to enter information that will be incorporated into
your certificate request. The information you enter is what is called a
Distinguished Name or a DN.
Country name (2 letter code) [US] US
State or province (full name) [Some state] TX
Locality name (city) [Some city] Dallas
Organization name (Company name) Brocade
Organizational unit name (department) Engineering
Common name (your domain name) www.brocade.com
Email address [webadmin@brocade.com] se@brocade.com
```

Syntax: ssl gencert certkey <*key-pair-file*> **signkey** <*key-pair-file*> <*password*> <*cert-name*>

The <key-pair-file> variable is the name of the RSA key pair used to build and sign this certificate. It is created using the **ssl genrsa** command.

The <password> variable is the password that is used to store this certificate.

The <cert-name> variable is the filename used to store the generated certificate. This file name can contain a maximum of 32 characters.

NOTE
To export a certificate off of a ServerIron ADX you need the **key-pair-file** and **password** configured here.

NOTE
To generate a self signed certificate, the certkey and sign key must be the same.

Using CA-signed certificates

Before generating a CA-signed certificate, you must obtain an RSA key pair as described in "Obtaining a ServerIron ADX keypair file" on page 140.

Once you've obtained the RSA key pair, you can create a certificate signing request (CSR) as shown in the following example.

```
ServerIronADX# ssl gencsr testkey
You about to be asked to enter information that will be incorporated into
your certificate request. What you are about to enter is what is called a
Distinguished Name or a DN.

Country name (2 letter code) [US] US
State or province (full name) [Some state] Texas
Locality name (city) [Some city] Dallas
Organization name (Company name) [Brocade Communications] Brocade
Communications
Organizational unit name (department) [Web administration] Engineering
Common name (your domain name) [www.brocade.com] www.brocade.com
Email address [webadmin@foundrynet.com] se@brocade.com
-----BEGIN CERTIFICATE REQUEST-----
MIIBszCCARwCAQAwczELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWEx
ETAPBgNVBAcTCFNhbiBKb3NlMRkwFwYDVQQKExBGb3VuZHJ5IG5ldHdvcmtzMQww
CgYDVQQLEwNTUUExEzARBgNVBAMTCnd3dy5xYS5uZXQwgZ8wDQYJKoZIhvcNAQEB
BQADgY0AMIGJAoGBANAF2D9/7qgQ+J9bPvVPhKkWP8GAQ3K4+aIRDYnizQon3unw
invhJImFCvmi+BFdYIj/aA5AQw0xI7618giZEL8VWGJ/D6j9U9wW3+x29QZnwqSF
QOyLRJia/gmpE8SMAeB88FLlZ0IZycEZUBZgolI9TJYnP4v6IM6ChceryCrrAgMB
AAGgADANBgkqhkiG9w0BAQUFAAOBgQBxI90CmjIqDtddLOQDCx5lVLCRnnC/rPnc
2gxu0SJM4eUVUD9DWW6exC9zvuXm3ZtS0CjEbrmorpoqnshEaPFb/tJJO9OQRZIW
CN6ZEdlwLNnh7M2xVdAdd9Nk4dqczM1n/nRD81tBS+rBP1mVwagKyiN2kiuDVCOs
ySuho58USQ==
-----END CERTIFICATE REQUEST-----
```

The output above, between BEGIN CERTIFICATE REQUEST and END CERTIFICATE REQUEST is a certificate signing request and registration. It is called a **Base-64-encoded CSR**.

The certificates that you import into the ServerIron will need to be in Base-64 encoding as well. The ServerIron accepts certificates in the .PEM and .P12 format. These formats are described further in this document as well as how to convert between formats.

You need to copy the certificate signing request mentioned above and register it with a trusted certificate authority; for example, Verisign. They will issue you a server certificate, based on this request (which will include the intermediate and root certificates).

You will then need to import these certificates into the ServerIron via SCP from a remote machine. Again, the certificates you import must be in Base 64 encoding. You can quickly tell if this is the case by opening the certificate in notepad. It should look like this:

```
-----BEGIN CERTIFICATE-----
```

```
MIIDKTCCApKgAwIBAgIRAJoKUHAGHghM4kW84LNXP1wwDQYJKoZIhvcNAQEFBQAw
ZDETMBEGCgmSJomT8ixkARkWA29yZzEYMBYGCgmSJomT8ixkARkWCGpvbmRhdmlz
MQ0wCwYDVQQKEwRUQU1VMREwDwYDVQQLEwhTZWN1cml0eTERMA8GA1UEAxMIVW5k
ZXJ0b3cwHhcNMDQwOTAyMTc1ODE3WhcNMDcwNzIzMTc1NzQxWjBkMRMwEQYKCZIm
iZPyLGQBGRYDb3JnMRgwFgYKCZImiZPyLGQBGRYIam9uZGF2aXMxDTALBgNVBAoT
BFRBTVUxETAPBgNVBAsTCFNlY3VyaXR5MREwDwYDVQQDEwhVbmRlcnRvdzCBnzAN
BgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAyk4jxC526rUPrkYC1pL+VobYp4B8yLEq
rzbYyL4G6g8OlQ5ZozfP8WHF0T9a7dr/2FmvzWNBka3mHgIUQQxVZcVe/4ALCSLU
tfHKaAWsgwzN+/86BFO6+/2ht2X0Yzo3laY69iGJAW1cNH/7DFE2sF42/EDk0VDb
mRU3cE4afOMCAwEAAaOB2jCB1zB3BgNVHR8EcDBuMGygaqBohwSsEAEFpDIwMDEu
MCwGA1UEAxMlb3U9U2VjdXJpdHksbz1UQU1VLGRjPWpvbmRhdmlzLGRjPW9yZ4Ys
aHR0cDovL3Njb3JwaW8uam9uZGF2aXMub3JnOjQ0Ny9VbmRlcnRvdy5jcmwwDgYD
VR0PAQH/BAQDAgGMAwGA1UdEwQFMAMBAf8wHwYDVR0jBBgwFoAUnGfclktn1nNL
ICknzxZsbFThFoEwHQYDVR0OBBYEFJxn3JZLZ9ZzSyApJ88WbGxU4RaBMA0GCSqG
SIb3DQEBBQUAA4GBAIg8VKUyiGCrQ4Rn6fRKQs4S1Paf6SPot5AQ1cHK5IlFHkFF
nUYMwFdQZcBrfXMLLPZb1ih0MtfEogLSbS82atF8VfkhzUAKl4ke7lKA35jr22Us
KhYtqbwzWkjBu4z/ph10L21CDSSghW1ea75+6IVEa/ZyuvOaINL2wQYNlmps
-----END CERTIFICATE-----
```

Syntax: ssl gencsr <*key-name*>

The <*key-name*> variable is the key name that you want to use for the certificate request.

Exporting Web Server Certificates

You can export a Web Server Certificate from a Web server and install it on a ServerIronADX. This section describes the procedures required to export Web server certificates from a Windows Internet Information server (IIS), or and Apache server (UNIX).

Windows IIS

To export an existing Web server certificate to install on a ServerIronADX, follow these steps:

1. In the **Run** dialog box, type **mmc**, and click **OK**. The Microsoft Management Console (MMC) appears.
2. If you do not have Certificate Manager installed in MMC, you need to install it. For more information on how to add the Certificate snap-in to an MMC console, see the Microsoft link: Install a Server Certificate.
3. In the console tree, click the logical store where the certificate you want to export exists. It is usually in the Certificates folder in the Personal directory under Certificates (local computer) on the console root.
4. Right-click the certificate you want to export and click **All Tasks** > **Export** to start the Certificate Export Wizard.
5. Click **Next**.
6. On **Export Private Key**, click **Yes** to export the private key.

   You must export the private key with your certificate for it to be valid on your target server. Otherwise, you must request a new certificate for the target server.
7. In the **Export File Format** dialog box, choose**.PFX**. If the certificate has already been formatted, that format is selected as the default. Click **Next**.

   Do not select **Delete the private key if export is successful**, because this disables the SSL site that corresponds to the private key.

8. Continue to follow steps in the wizard, and enter a password for the certificate backup file when prompted. Using a strong password is highly recommended to ensure that the private key is well protected.
9. Type the name of the file you want to export, or click **Browse** to search for the file. Click **Next**.
10. Click **Finish** to exit the Certificate Export Wizard.

In order for certificates to be imported into the ServerIron ADX, they must be in a specific format. The .PFX file must be converted to .PEM or .P12. Procedures for converting certificate formats are described in "Converting certificate formats" on page 147.

11. When prompted for the import password, enter the password you used when exporting the certificate to a PFX file. You should receive a message that says **MAC verified OK**. The resulting file contents will resemble the following:

```
    1.3.6.1.4.1.311.17.2: <No Values>
    localKeyID: 01 00 00 00
    Microsoft CSP Name: Microsoft RSA SChannel Cryptographic Provider
    friendlyName: 740b399c4eb957ca6b972da9345dbda3_e7adefb8-6420-4ed2-
b6f5-2c4988094b69
Key Attributes
X509v3 Key Usage: 10
-----BEGIN RSA PRIVATE KEY-----
MIICWwIBAAKBgQCrSx5QmEBB35QZqKZLzt9uC9ZPmL8sNI/yNfDlh0BnT3Y0xcxV
Uo+buHIwGNcFKQgTQJRvZ1UlvLfKzETT9VWlUv6PsWLonxUZEWFion+ThOo5zQYe
/i/EZe5APF0CKdBAVlBONs2FOsxztTA2zrBzpxtlc7NyjJWLmYdjqyQujwIDAQAB
AoGAbugfQ62ghUY+vPJOYe0bC0SMyVKsDPIndXrpc6PD1mBDt92N9HyVUQz2mh+r
MMMlYFnDAfKU1e8/zQW/VsGVZnuIVgc5kk43BGo69o5Tvvw/7w+Vr8X95fXie72i
noWPvaOsONoV72oBKsKCfxgmEwurGWT/MoL6KG0Uw65bCLECQQDUYYg5R7cS9pKi
bX8yVv71l5gazYbo34YqDXniQlY2PtxTKjkg4cJ1WxbJeSRAj7N+kteKsp0emBJ8
OyeOt+UJAkEAznlN3YYn+WTM2whGfivj/B+EkMaB0sJJNAnWg4Zd1aR40uQsnNVE
Qj6svvFAYNY9zEZ5XG4Kbcl4zilH4wU01wJAHCNr9ALfa6DFnsWZI6dJEmrqWSq/
5ByxXuL4MhUN7RP5Bv/CH+wpYnkWFM17Ex627l8wNY0iqgoF8PW5cPxGWQJAIswU
0GG8uO5BDCT/5GG3QZV6PQAHfyKZPbjuuynZBJl8d2ZXEnNq1pNDf8ae5/MTZSr3
ZxlwjoIm5u/UGrGU7wJAR4lxl9/qhu+db0aBjBIEOfs/9NuRh5RMDEZF+ZXdKcE9
inV5o51vyVLMXZfknishbKWOBS62POMXJsOt+w9I0g==
-----END RSA PRIVATE KEY-----
Bag Attributes
    localKeyID: 01 00 00 00
    friendlyName: www.brocade.com
subject=/C=US/ST=Texas/L=Dallas/O=Brocade/OU=Engineering/CN=www.brocade.com
issuer=/DC=com/DC=brocade/O=Brocade/OU=Engineering/CN=www
-----BEGIN CERTIFICATE-----
MIICxTCCAi6gAwIBAgIQDmM2IYR4YnLzCLr31Z/DPzANBgkqhkiG9w0BAQUFADBm
MRMwEQYKCZImiZPyLGQBGRYDb3JnMRgwFgYKCZImiZPyLGQBGRYIam9uZGF2aXMx
EDAOBgNVBAoTB1Rla2VsZWMxETAPBgNVBAsTCFNlY3VyaXR5MRAwDgYDVQQDEwdU
ZWtlbGVjMB4XDTA1MDQxOTAxNTY0OFoXDTA3MDgwNDE3NDQ0OFowczELMAkGA1UE
BhMCVVMxDjAMBgNVBAgTBVRleGFzMQ8wDQYDVQQHEwZEYWxsYXMxEDAOBgNVBAoT
B1Rla2VsZWMxETAPBgNVBAsTCFNlY3VyaXR5MR4wHAYDVQQDExVhaXJlc3BhY2Uu
dGVrZWxlYy5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAKtLHlCYQEHf
lBmopkvO324L1k+Yvyw0j/I18OWHQGdPdjTFzFVSj5u4cjAY1wUpCBNAlG9nVSW8
t8rMRNP1VaVS/o+xYuifFRkRYWKif5OE6jnNBh7+L8Rl7kA8XQIp0EBWUE42zYU6
zHO1MDbOsHOnG2Vzs3KMlYuZh2OrJC6PAgMBAAGjZzBlMBMGA1UdJQQMMAoGCCsG
AQUFBwMBMA4GA1UdDwEB/wQEAwIDODAfBgNVHSMEGDAWgBRW7ldC6sXhjQlDL+Sv
jLh4fOG5wTAdBgNVHQ4EFgQUG1P0iW0rJX1rUYtlEFmzayzxHkEwDQYJKoZIhvcN
AQEFBQADgYEAgI7MP/4PSiyDm/vZzE4Rmxlhrju+6xN9Q2PVVpOY3P/WE3QOtkOw
4nShKlLucDaC0m+sJjKzVi5Ezv7ebWSKq5rF+3NvgOIouh2vqCI1AgyI7QCkxTWW
PoYSssBzh+b+IK6+XsbekTfH3ghcM1+Yx+RZP9eIGSFRT/oTZnC87D0=
-----END CERTIFICATE-----
```

```
Bag Attributes: <Empty Attributes>
subject=/DC=org/DC=test/O=root/OU=Security/CN=root
issuer=/DC=org/DC=test/O=root/OU=Security/CN=root
-----BEGIN CERTIFICATE-----
MIIC1TCCAj6gAwIBAgIQJhB5wR9FdbXPEWcLp/1MAjANBgkqhkiG9w0BAQUFADBm
MRMwEQYKCZImiZPyLGQBGRYDb3JnMRgwFgYKCZImiZPyLGQBGRYIam9uZGF2aXMx
EDAOBgNVBAoTB1Rla2VsZWMxETAPBgNVBAsTCFNlY3VyaXR5MRAwDgYDVQQDEwdU
ZWtlbGVjMB4XDTA1MDQxOTAxMTk1OFoXDTA3MDgwNzE3NDM1OFowZjETMBEGCgmS
JomT8ixkARkWA29yZzEYMBYGCgmSJomT8ixkARkWCGpvbmRhdmlzMRAwDgYDVQQK
EwdUZWtlbGVjMREwDwYDVQQLEwhTZWN1cml0eTEQMA4GA1UEAxMHVGVrZWxlYzCB
nzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAq36AVcI33Pp9tPjuN2Dx81BIiUTx
ZENHS/0ZL4RREj+BfZG3/J94cE0i5F6l0X9W6jJpUM8sqUVqpounwB6ZeoXHJsQJ
Hvzp1YR77ABS1gR//b9N3TiIXGyb8oaoXdT4xahzfwMTTjOGAGl3xYHC/QdXi3x6
ff+X02AddhIvhaMCAwEAAaOBgzCBgDAMBgNVHRMEBTADAQH/MCAGA1UdJQEB/wQW
MBQGCCsGAQUFBwMBBggrBgEFBQcDAjAOBgNVHQ8BAf8EBAMCAQYwHwYDVR0jBBgw
FoAUVu5XQurF4Y0JQy/kr4y4eHzhucEwHQYDVR0OBBYEFFbuV0LqxeGNCUMv5K+M
uHh84bnBMA0GCSqGSIb3DQEBBQUAA4GBAFCldN7DHtztK2hdiUp1KO1LtO9Ics9g
mjVH869i6qxVOj+YzGfBlz7PvNdW+Nv0TCrrXTLXgZpd1aAW/lTajBfLgFs21Xkf
xSquYFYEcZjz4Uu3gMuuAiS963Xissy+MIyNJpkRP1NpYY75lXAf05sLopzcmdVc
C2es4LOJQbhZ
-----END CERTIFICATE-----
```

12. You can now begin copying the certificates and the key pair files to the ServerIronADX (in the following order):

```
scp ./server-key.pem admin@192.168.1.1:sslkeypair:server-key:foundry:pem
scp ./server-cert.cer admin@192.168.1.1:sslcert:certchain1:pem
```

Unix (Apache)

The following procedure describes the procedure for determining the location of the private key and certificate files and copying them to a ServerIronADX.

1. On the Apache server, look in the httpd.config file for the following directives; they point to the location of the key and certificate files:

```
SSLCertificateFile .../path/to/mycertfile.crt
SSLCertificateKeyFile .../path/to/mykeyfile.key
```

NOTE
The default location of the httpd.config file is: /etc/httpd/conf/httpd.conf

2. When you have located the key and certificate files, copy them from the Linux server to the ServerIronADX:

```
scp ./server-key.key admin@192.168.1.1:sslkeypair:server-key:foundry:pem
scp ./server-cert.crt admin@192.168.1.1:sslcert:certchain1:pem
scp ./root-cert.crt admin@192.168.1.1:sslcert:certchain1:pem
```

Make sure you upload in the same order as the CA hierarchy – only then can the chain be established properly on the ServerIron.

NOTE
You must upload all of the chain certificates into the same file on the ServerIronADX.

Converting certificate formats

The ServerIronADX accepts server certificates in the PEM or PKCS12 format. The following sections describe how to convert between the two formats and from PFX to the two formats using OpenSSL. You can download a Win32 distribution of OpenSSL at the following location:

http://gnuwin32.sourceforge.net/packages/openssl.htm

Converting PEM to PKCS12

Use the open-source utility OpenSSL to perform the conversion from .PEM to .P12 as described in the following

1. If you do not have it installed, download and install the Win32 OpenSSL package from the URL described under "Converting certificate formats".
2. Create a folder named C:\certs and copy the file pemkey.pem (private key file) and pemcert.pem (certificate file) into this folder
3. Open a command prompt and move to the GnuWin32\bin directory:

   ```
   cd %ProgramFiles%\GnuWin32\bin
   ```

4. Enter the following command to convert the PEM file to an P12 file (all on one line):

   ```
   openssl pkcs12 –export –out c:\certs\mycert.p12 –name "My Certificate" –inkey
   c:\certs\pemkey.pem –in c:\certs\pemcert.pem
   ```

5. When prompted for the import password, enter the password you used when exporting the PEM certificate.
6. When prompted for a pass phrase, enter a password you want to use to protect the private key.

Converting PKCS12 to PEM

To convert a P12 file to a PEM file, follow these steps on a Windows machine:

1. If you do not have it installed, download and install the Win32 OpenSSL package from the URL described under "Converting certificate formats".
2. Create a new folder named C:\certs and copy the file pkcsfile.p12 into this folder
3. Open a command prompt and move to the GnuWin32\bin directory.

   ```
   cd %ProgramFiles%\GnuWin32\bin
   ```

4. 4. Enter the following command to convert the P12 file to an PEM file:

   ```
   openssl pkcs12 -in c:\certs\pkcsfile.p12 -clcerts -out c:\certs\pemfile.pem
   ```

5. When prompted for the import password, enter the password you used to export the PEM certificate.
6. When prompted for a pass phrase, enter a password to protect the private key. This creates a single .PEM file that includes both the private key and certificate.
7. You can then separate the private key and certificate into separate .PEM files.

Use the open-source utility OpenSSL to perform the conversion from .PFX to .PEM (or .PFX to .P12). You can download a Win32 distribution of OpenSSL at the following location:

http://gnuwin32.sourceforge.net/packages/openssl.htm

Converting a PFX file to a P12 file

To convert a PFX file to a P12 file on a Windows machine, change the extension from .PFX to .P12

Converting a PFX file to a PEM file

To convert a PFX file to a PEM file on a Windows machine, follow these steps:

1. If you do not have it installed, download and install the Win32 OpenSSL package from the URL described under "Converting certificate formats".
2. Create a folder C:\certs and copy the file yourcert.pfx into the c:\certs folder
3. Open a command prompt and change into the GnuWin32\bin directory:

   ```
   cd %ProgramFiles%\GnuWin32\bin
   ```

4. Type the following command to convert the PFX file to an unencrypted PEM file (all on one line):

   ```
   openssl pkcs12 -in c:\certs\yourcert.pfx -out c:\certs\yourcert.pem –nodes
   ```

Importing keys and certificates

You can import keys and certificates to and from a ServerIron ADX. Generally you export a CSR (certificate signing request) from a ServerIron ADX, have it signed by a CA, and import it back into the ServerIron ADX.

You also need the import and export functionality if you want to use client-authentication to transfer the CA certificate to a ServerIron ADX.

The ServerIron ADX supports SCP protocol for transferring keys and certificates. An internal SCP server is provided and must be enabled before any transfer. A ServerIron ADX only responds to transfer requests initiated by remote clients.

Enabling an SCP server

The SCP server relies on the SSH protocol. SCP is enabled by default. But, you need to enable SSH first to generate the SSH key.

If the SSH server is not enabled, take the following steps to enable it.

1. Configure a domain name as shown. In this example, "www.mydomain.com" is configured as the domain name.

   ```
   ServerIronADX(config)# ip dns domain-name www.mydomain.com
   ```

2. Create a username. In this example, "secret" is configured as the username.

   ```
   ServerIronADX(config)# username admin password secret
   ```

3. Enable SCP.

   ```
   ServerIronADX(config)# crypto key generate dsa
   ```

4. Allow empty SSH passwords.

   ```
   ServerIronADX(config)# ip ssh permit-empty-passwd ye
   ```

Detailed descriptions of the commands used in this procedure are described in the following manual: *ServerIron ADX Administration Guide*.

Windows Users

GUI-based SCP tools do not work in the current environment when you use SCP to transfer the certificate files to the ServerIronADX. Windows users should have PSCP, a free SCP utility based on putty SSH client. To access this Windows utility, use the following commands:

```
C:\images>pscp first.cer admin@200.100.100.2:sslcert:bs:pem
C:\images>pscp second.cer admin@200.100.100.2:sslcert:bs:pem
```

To upload a key-pair to a ServerIron ADX:

Syntax: pscp <key-pair-file-name>
<user>@<SI_IP_Addr>:sslkeypair:<filename-on-SI>:<password>:<format>

To download a key-pair from a ServerIron ADX:

Syntax: pscp <user>@<SI_IP_Addr>:sslkeypair:<filename_on_SI>:<password>:<format>
<key-pair-file-name>

To upload a certificate file to a ServerIron ADX:

Syntax: pscp <cert-file-name> <user>@<SI_IP_Addr>:sslcert:<filename-on-SI>:<format>

To download a certificate file from the ServerIron ADX:

Syntax: pscp <user>@<SI_IP_Addr>:sslcert:<filename-on-SI>:<format> <cert-file-name>

Example:

The following example uploads a certificate file named: "first.cert" to a ServerIron ADX and saves it with the name "bs" in pem format:

```
C:\images>pscp first.cer admin@200.100.100.2:sslcert:bs:pem
```

Transferring a Keypair File and a Certificate File

For a ServerIron ADX to recognize the incoming file type, the filename must be in a specific format. With the correct format, you can describe the file type, file name, password, and format.

The name is divided into fields, which are separated by colons (:). The following fields are used:

- File type - Determines whether the file contains a key pair or a certificate. The **sslcert** keyword is used for a certificate. The **sslkeypair** keyword is used for a key pair.
- File name - The file name on the ServerIron ADX flash. The name cannot be more than 25 characters for the key pair file and 32 characters for the certificate file.
- Password - Only required for a keypair file password. It is not used in certificate files. The password cannot be more than 64 characters.
- Format - Describes the format of the file. It can either be **pem** or **pkcs12**.

Based on these rules, use the following syntax to upload a file to ServerIronADX:

Syntax: **scp** *<source-file> <username>* **@***<SI_IP_Addr>:<filetype>:<filename>:<password>:<format>*

NOTE: You do not need the password field for PEM format certificate files.

NOTE

For example, if a keypair file, "myrsakeys" needs to be uploaded to ServerIron ADX, its password is "brocade," and it is in PEM format. The SCP server is already enabled on the ServerIron ADX and a user "admin" is also created.

To configure this scenario, use the following command:

```
c:\ scp myrsakeys.pem admin@<ip_addr>:sslkeypair:myrsakeys:brocade:pem
```

After uploading the keypair file, the same file can be downloaded to a client with the following command:

```
c:\ scp admin@<ip_addr>:sslkeypair:myrsakeys:foundry:pem myrsakeys.pem
```

NOTE
The downloaded file includes the following additional block of text at the end.
----BEGIN RSA PUBLIC KEY-----
MIGJAoGBANY8/gNKT42GTweT+/c34CRxRwmaUvQQbTMgxYhHdLbo1g+6sdDcrohH
IlXVOWJH4pjt9JB1zFaM/rSBnvKGkJ67HbT7dvszQnLNtg9aZnX3i5vPjFhjm9mj
j9E9alg/3CD1GpOXH40BJBZ3F8HFYaH8EOLlp5gLf/hxAYTPDQ2DAgMBAAE=
-----END RSA PUBLIC KEY-----
This additional block of text are the public key associated with the certificate, which does not create any issues.

Similarly, a certificate file can be uploaded to or downloaded from the ServerIron ADX. For example, to upload the certificate file "mycertfile" to the ServerIron ADX, which is in PEM format, use the following command:

```
c:\scp mycertfile admin@<ip_addr>:sslcert:mycertfile:pem
```

NOTE
There is no password field.

To download the same file from the ServerIron ADX back to the client, use the following command:

```
c:\ scp admin@<ip_addr>:sslcert:mycertfile:pem
```

In general, use the following commands:

- To upload a key-pair to a ServerIron ADX:

Syntax: scp *<key-pair-file-name><user>***@***<SI_IP_Addr>***:sslkeypair:***<filename-on-SI>***:***<password>:<format>*

- To download a key-pair from ServerIronADX:

Syntax: scp *<user>***@***<SI_IP_Addr>***:sslkeypair:***<filename_on_SI>***:***<password>***:***<format> <key-pair-file-name>*

- To upload a certificate file to the ServerIronADX:

Syntax: scp *<cert-file-name><user>***@***<SI_IP_Addr>***:sslcert:***<filename-on-SI>***:***<format>*

- To download a certificate file from the ServerIronADX:

Syntax: scp *<user>***@***<SI_IP_Addr>***:sslcert:***<filename-on-SI>***:***<format><cert-file-name>*

When a key-pair file or a certificate file is uploaded, you can view it using the **show ssl cert** command described in "Displaying locally stored SSL certificates" on page 190.

Additional Notes for PKCS12

PKCS12 format stores keys and certificates in the same file. You must use the scp keyword **keypairfile** and command syntax of keypairfile while transferring a PKCS#12 file to the ServerIronADX.

To transfer a certificate and key file in PKCS format (mypkcsfile) to a ServerIron ADX, use the following command:

```
c:\ scp ./mypkcsfile.p12 admin@<ip_addr>:sslkeypair:mypkcsfile:brocade:pkcs12
```

After transferring the file, it can be used both as a key and a certificate. To add the certificate file and keys to the profile, use the following commands:

```
ServerIronADX(config-ssl-profile-mysslprofile)# keypair-file mypkcsfile
ServerIronADX(config-ssl-profile-mysslprofile)# certificate-file mypkcsfile
```

The **show ssl cer**t command can be used to display a pkcs file. The **show ssl key** command does not display a pkcs file, but it does contain a keypair.

Creating a Master Password for export of SSL keys

You can create a master password that grants permission to export all SSL keys on a ServerIron ADX using SCP copy. This password is used with the "**scp** <*key-pair-file-name*> <*user*>**@**<*SI_IP_Addr*>**: sslkeypair:**<*filename-on-SI*>**:**<*password*>*: <format>"* command. If a master password is not configured, a separate password associated with each key must be used.

To define a master password for the export of SSL keys, use the following command.

```
ServerIronADX# ssl set export-master-pswd exportpw
```

Syntax: ssl set export-master-pswd <password>

The **<password>** variable specifies the master password for export of SSL keys. It can be from 1 to 24 characters in length.

To disable a master password for export of SSL keys, use the following command.

```
ServerIronADX# ssl clear export-master-pswd exportpw
```

Syntax: ssl clear export-master-pswd <password>

The **<password>** variable specifies the master password for export of SSL keys that you want to remove.

Use the following command to display whether a master password is in effect.

```
ServerIronADX# show ssl key *
master-password enable
ssl  key  files:
1   : key-test
2   : key1
3   : keyz
4   : keyc
5   : key7
```

**Syntax: show ssl key *

Deleting certificate and key files

You can use the following commands to delete a specified certificate or key file as shown.

```
ServerIronADX# ssl clear certfile <certfile-name>
```

Syntax: ssl clear certfile <*certfile-name*>

The <certfile-name> variable specifies the certificate that you want to delete.

```
ServerIronADX# ssl clear keyfile <keyfile-name>
```

Syntax: ssl clear certfile <*keyfile-name*>

The <keyfile-name> variable specifies the key that you want to delete.

Certificate Verification

Every certificate has two very important fields: issuer (issued-by) and subject (issued-to). A CA's certificate has the same value in both fields, because the authority has issued a certificate to itself. However, when the authority issues a certificate to a server, the issuer field contains the CA's name, but the subject contains the server's name.

For example, the following server certificate was issued by Verisign (a CA):

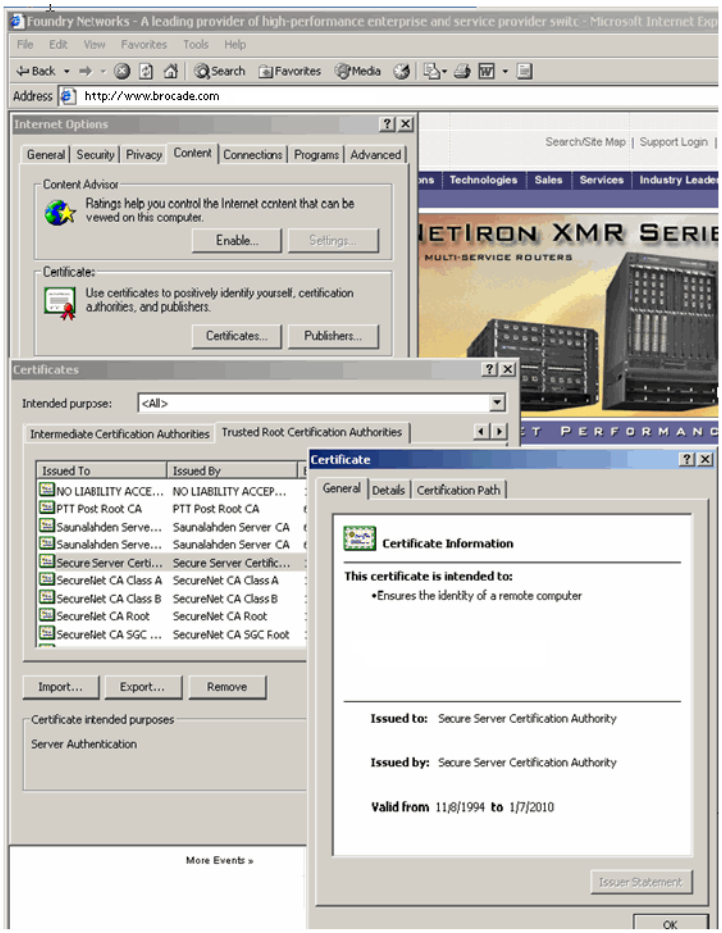
```
Issuer: C=US, O=RSA Data Security, Inc., OU=Secure Server Certification Authority
Subject: C=US, ST=California, L=San Jose, O=Brocade Inc, OU=L47 and Security
Group, OU=Terms of use at www.verisign.com/rpa (c)05, CN=l47qa.foundrynet.com
```

To authenticate this server certificate, the client, for example, Firefox or IE, should have the corresponding CA's certificate. When you open the trusted root CA page in Internet Explorer, you can also see that entry has the same value in the issued by (issuer) and issued to (subject) fields.

This is an example of how a server certificate is issued directly by a CA. Note that in this scenario, the server sends only its own certificate and not that of the CA.

Figure 11 shows a CA certificate.

FIGURE 11 Certificate



Chained Certificate Verification

When the server certificate is not signed directly by the root CA, but signed by an intermediate CA, as shown in the following example, there are two possible scenarios.

- CA ----> intermediate CA ----> server certificate

Client Already Has Intermediate CA's Certificate

In the first scenario, there are NO additional requirements. When the server sends a certificate that is signed by the intermediate CA, the client browser will be able to process it successfully.

Client Does NOT Have Intermediate CA's Certificate

In the second scenario, the server sends a certificate that is signed by intermediate CA. However since the end-client has no knowledge of the intermediate CA, it denies the certificate and the process is unsuccessful.

To resolve this issue, the server must send not only its own certificate, but also the intermediate CA's certificate that is signed the root CA. In other words, the server sends a chain of certificates.

NOTE
The server sends only its own certificate and the intermediate CA's certificate. It does NOT send the root CA's certificate.

Example

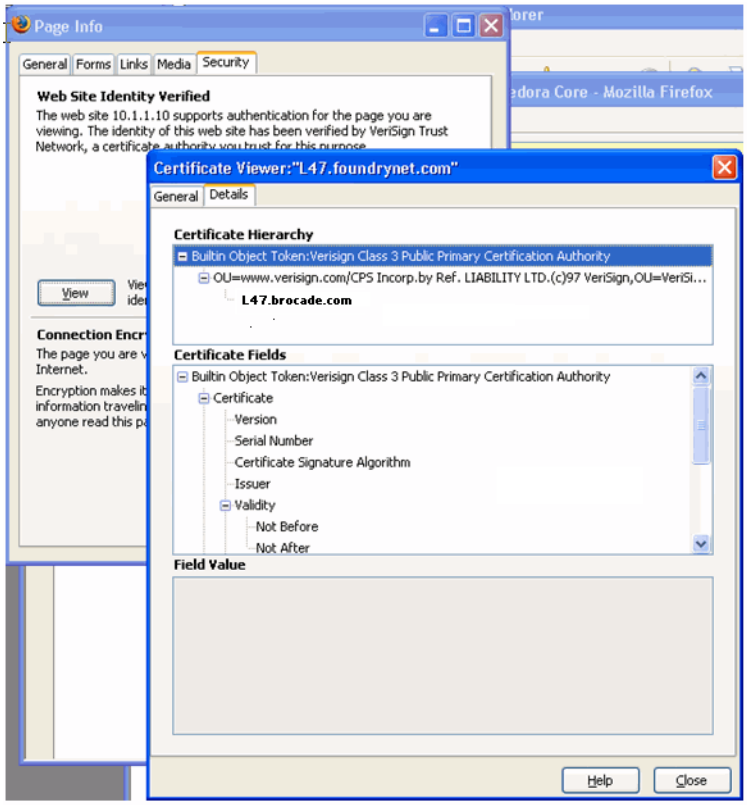
Your server certificate is signed by VeriSign International Server CA - Class 3. This is an intermediate CA, whose certificate is signed by VeriSign Class 3 Public Primary CA.

Figure 12 shows the certificate hierarchy, with "L47.brocade.com" at the third level. The first level certificate is also labeled as "built-in object token" by Firefox. This is an example of chaining. The server sends a two-level chain containing its own certificate and the certificate of the intermediate CA.

The certificate chain sent by the server must be correct: server-> intermediate CA. The intermediate CA certificate must also be signed by a CA whose certificate is present with the client.

Figure 12 shows the certificate fields.

FIGURE 12 Certificate Fields



There are two steps that will ensure that the chain is correct.

1. Verify that the issuer of the server certificate matches the subject of the intermediate CA's certificate.
2. Verify that the issuer of the intermediate CA's certificate has an entry in the client's trusted certificates.

   For the first step, you must convert the certificate chain to a readable format. From the BP console, use the **show ssl cert** <cert chain name> command to convert the chain to readable format. All certificates in the chain are displayed and every certificate must begin with the keyword **certificate**.

```
ServerIronADX# show ssl cert *
1   :verisign128cert
2   :cert2112.pem
3   :cert2031.pem
4   :cert4030
3   :cert2031


ServerIronADX# show ssl cert verisign128cert
Certificate:
       Dat        Version: lu (0xlx)
```

```
        Serial Number:
            70:2b:a7:4b:07:ea:29:99:5a:dc:3f:6f:74:da:39:6d
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: O=VeriSign Trust Network, OU=VeriSign, Inc., OU=VeriSign
International Server CA - Class 3, OU=www.verisign.com/CPS Incorp.by Ref.
LIABILITY LTD.(c)97 VeriSign
        Validity
            Not Before: Nov  2 00:00:00 2005 GMT
            Not After : Nov  2 23:59:59 2006 GMT
        Subject: C=US, ST=California, L=San Jose, O=Brocade Inc, OU=Engineering,
OU=Terms of use at www.verisign.com/rpa (c)05, CN=L47.brocade.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:a6:10:cc:70:dd:36:28:06:3c:c0:5f:c7:c0:44:
                    a6:54:cd:fc:2d:e3:a6:68:50:11:03:e5:e7:16:97:
                    f3:ba:b1:42:de:d0:df:26:f6:35:8f:22:a1:a4:3d:
                    fd:7a:35:08:ed:8c:5f:6c:ab:ca:13:1f:87:a0:c4:
                    dd:30:ea:00:18:b8:2b:24:25:13:60:c1:08:e4:af:
                    da:25:a9:e0:ef:c3:34:13:41:02:b2:39:83:1a:49:
                    bd:95:4e:29:3e:e9:a1:a4:d1:f9:0d:d1:80:3d:01:
                    ff:af:d2:a8:00:6a:2a:e2:97:cd:f5:5c:24:a4:88:
                    a2:a2:6f:da:1a:0d:8f:fa:f7
                Exponent: lu IÕ8~0xlx)
*s:
*sX509v3 Basic Constraints:
*sCA:FALSE
*sX509v3 Key Usage:
*sDigital Signature, Key Encipherment
*sX509v3 Certificate Policies:
*sPolicy: 2.16.840.1.113733.1.7.23.3
*sCPS:

*sX509v3 CRL Distribution Points:
*sURI:http://SVRIntl-crl.verisign.com/SVRIntl.crl

*sX509v3 Extended Key Usage:
*sTLS Web Server Authentication, TLS Web Client Authentication, Netscape Server
Gated Crypto
*sAuthority Information Access:
*sOCSP - URI:http://ocsp.verisign.com
*sCA Issuers - URI:http://SVRIntl-aia.verisign.com/SVRIntl-aia.cer

*s1.3.6.1.5.5.7.1.12:
*s0_.].[0Y0W0U..image/gif0!0.0...+.............k...j.H.,{..0%.#http://logo.veris
ign.com/vslogo.gif
    Signature Algorithm: sha1WithRSAEncryption
        c8:65:15:64:42:ea:36:f4:d4:68:c4:ad:b9:1f:d9:03:fb:d9:
        15:27:1b:f6:a2:e0:ea:ae:74:1e:de:94:17:36:0f:63:19:8f:
        34:bf:f1:32:02:d5:c5:79:0d:bf:f8:56:62:34:67:4d:ad:b8:
        40:e1:51:4f:2d:28:32:7f:20:ad:19:53:6b:6b:9e:c6:c1:50:
        9e:89:fb:c1:f1:33:88:36:64:8a:28:d1:c3:1f:b4:c8:f8:c3:
        af:5a:f2:77:86:67:3b:28:bb:84:17:a0:48:46:18:9b:f2:25:
        a1:e3:74:f6:34:08:f0:ed:68:65:e5:89:27:07:94:df:0c:9b:
        81:f5
Certificate:
       Dat        Version: lu (0xlx)
        Serial Number:
            25:4b:8a:85:38:42:cc:e3:58:f8:c5:dd:ae:22:6e:a4
```

```
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, O=VeriSign, Inc., OU=Class 3 Public Primary Certification
Authority
        Validity
            Not Before: Apr 17 00:00:00 1997 GMT
            Not After : Oct 24 23:59:59 2011 GMT
        Subject: O=VeriSign Trust Network, OU=VeriSign, Inc., OU=VeriSign
International Server CA - Class 3, OU=www.verisign.com/CPS Incorp.by Ref.
LIABILITY LTD.(c)97 VeriSign
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:d8:82:80:e8:d6:19:02:7d:1f:85:18:39:25:a2:
                    65:2b:e1:bf:d4:05:d3:bc:e6:36:3b:aa:f0:4c:6c:
                    5b:b6:e7:aa:3c:73:45:55:b2:f1:bd:ea:97:42:ed:
                    9a:34:0a:15:d4:a9:5c:f5:40:25:dd:d9:07:c1:32:
                    b2:75:6c:c4:ca:bb:a3:fe:56:27:71:43:aa:63:f5:
                    30:3e:93:28:e5:fa:f1:09:3b:f3:b7:4d:4e:39:f7:
                    5c:49:5a:b8:c1:1d:d3:b2:8a:fe:70:30:95:42:cb:
                    fe:2b:51:8b:5a:3c:3a:f9:22:4f:90:b2:02:a7:53:
                    9c:4f:34:e7:ab:04:b2:7b:6f
                Exponent: lu IÕ8~0xlx)
*s:
*sX509v3 Basic Constraints:
*sCA:TRUE, pathlen:0
*sX509v3 Certificate Policies:
*sPolicy: 2.16.840.1.113733.1.7.1.1
*sCPS:

*sX509v3 Extended Key Usage:
*sTLS Web Server Authentication, TLS Web Client Authentication, Netscape Server
Gated Crypto, 2.16.840.1.113733.1.8.1
*sX509v3 Key Usage:
*sCertificate Sign, CRL Sign
*sNetscape Cert Type:
*sSSL CA, S/MIME CA
*sX509v3 CRL Distribution Points:
*sURI:http://crl.verisign.com/pca3.crl
    Signature Algorithm: sha1WithRSAEncryption
        08:01:ec:e4:68:94:03:42:f1:73:f1:23:a2:3a:de:e9:f1:da:
        c6:54:c4:23:3e:86:ea:cf:6a:3a:33:ab:ea:9c:04:14:07:36:
        06:0b:f9:88:6f:d5:13:ee:29:2b:c3:e4:72:8d:44:ed:d1:ac:
        20:09:2d:e1:f6:e1:19:05:38:b0:3d:0f:9f:7f:f8:9e:02:dc:
        86:02:86:61:4e:26:5f:5e:9f:92:1e:0c:24:a4:f5:d0:70:13:
        cf:26:c3:43:3d:49:1d:9e:82:2e:52:5f:bc:3e:c6:66:29:01:
        8e:4e:92:2c:bc:46:75:03:82:ac:73:e9:d9:7e:0b:67:ef:54:
        52:1a
```

Once the chain is verified, the second step is to make sure that the intermediate CA is signed by a root CA whose certificate already exists in the client.

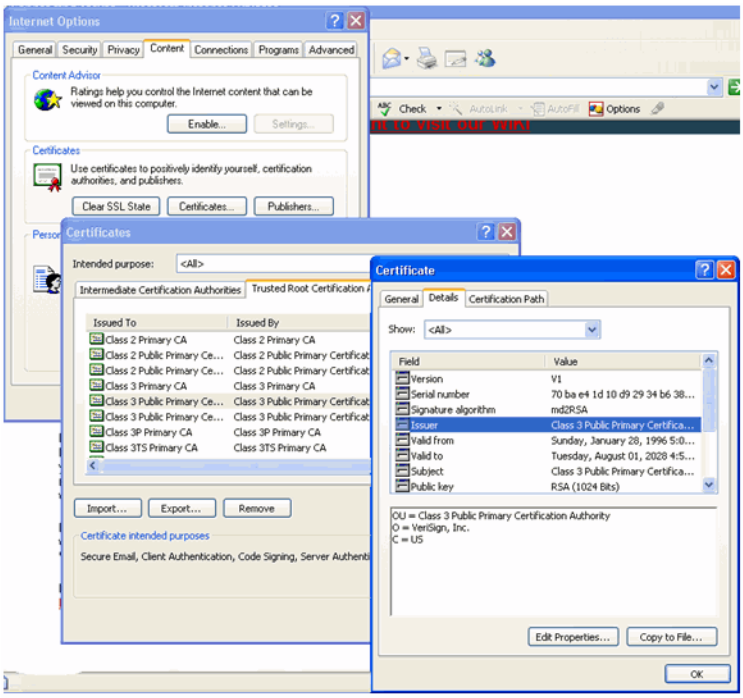
In this example, the intermediate certificate is signed by:

Issuer: C=US, O=VeriSign, Inc., OU=Class 3 Public Primary Certification Authority

Find and match this certificate in the list of trusted root certificates from the client browser.

Figure 13 shows the issuer certificate authority window.

FIGURE 13 Issuer Certificate Authority



Now the certificate chain is complete and the client browser will able to interpret it correctly. Let's consider another example with four level chain. Here, the root Certificate Authority is called as "OS Level_0 CA". This CA has two intermediaries:

- "OS Level_1 CA" and "OS Level_2 CA"

The second level intermediate CA has signed a server certificate with the following CN "ServerCert by Level_2":

`OS Level_0 CA à OS Level_1 CAà OS Level_2 CA à ServerCert`

The certificate hierarchy is shown as under:

```
Level 0 (root)  issuer : CN=OS Level_0 CA
                         Subject : CN=OS Level_0 CA
Level 1 (first intermediary: Issuer : CN=OS Level_0 CA
                         Subject : CN=OS Level_1 CA
Level 2 (Second intermediary:Issuer : CN=OS Level_1 CA
                         Subject: CN=OS Level_2 CA
Level 3 (Server Certificate)Issuer: CN=OS Level_2 CA
                         Subject: CN=ServerCert by Level_2
ServerIronADX# show ssl cert l4chaincert
Certificate:
       Dat        Version: lu (0xlx)
        Serial Number: 3 (0x00000003)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: CN=OS Level_2 CA
        Validity
            Not Before: Feb 10 03:14:21 2006 GMT
            Not After : Feb  8 03:14:21 2016 GMT
        Subject: CN=ServerCert by Level_2, O=Foundry Nets, OU=L47QA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:bb:d1:5d:8d:5a:ac:0e:94:ec:6c:49:fa:0e:03:
                    cd:c1:84:52:f0:e6:be:5d:a8:d3:36:c0:33:19:67:
                    d9:d0:1a:27:87:68:ce:06:68:b1:35:53:64:01:27:
                    67:4a:69:6d:1f:6f:2e:99:0a:f2:85:ea:fb:1f:f0:
                    99:21:26:ff:f5:50:11:22:a6:55:cd:fa:b1:2f:be:
                    5d:cf:65:be:4d:1e:37:e1:64:46:69:c1:73:e5:de:
                    d5:1d:09:ef:f0:e7:fa:c3:b5:f1:90:21:d5:84:23:
                    24:8e:9d:f7:35:66:7e:c0:97:af:61:ee:5a:3e:31:
                    b6:a7:5f:b9:81:1d:0d:43:d9
                Exponent: lu IÕ8~0xlx)
*sNetscape Cert Type:
*sSSL Server
*sX509v3 Key Usage:
*sDigital Signature, Non Repudiation, Key Encipherment
*sX509v3 Extended Key Usage:
*sTLS Web Server Authentication
*sNetscape CA Revocation Url:
*s
*sX509v3 Subject Key Identifier:
*s
*sX509v3 Authority Key Identifier:
*skeyid:23:77:98:42:E1:C1:BC:E7:9A:92:79:8E:DF:8D:C3:C1:2A:35:F2:0F
*sDirName:/CN=OS Level_1 CA
*sserial:01

*sAuthority Information Access:
*sCA Issuers - URI:http://s1.l47qa.com/l2/ca.crt

*sX509v3 CRL Distribution Points:
*sURI:http://s1.l47qa.com/l2/crl-v2.crl
*sX509v3 Certificate Policies:
*sPolicy: 1.1.1.1.1
```

```
*sX509v3 Certificate Policies:
*sPolicy: 1.1.1.1.1
*sCPS:
*sUser Notice:
*sExplicit Text:

*sX509v3 Issuer Alternative Name:
*semail:root@s1.l47qa.com, URI:http://sq.l47qa.com
*sX509v3 Subject Alternative Name:
*s<EMPTY>

   Signature Algorithm: sha1WithRSAEncryption
        8f:e0:08:8b:ea:69:9e:6b:45:d1:ef:e1:d0:ae:f5:74:9f:b7:
        98:1a:83:fa:95:72:bf:d9:0c:91:b0:c4:e9:0a:e6:08:20:eb:
        88:d9:b1:79:92:85:ce:26:6a:d5:31:d2:40:39:94:f0:58:6e:
        29:24:ba:c8:f1:b0:dc:d9:80:c9:25:42:68:fa:e1:04:5b:e0:
        c4:98:c9:61:97:2b:49:a8:74:ea:31:ee:7b:ec:ae:f0:8f:20:
        32:b5:27:35:e0:dc:71:61:ed:ca:eb:31:bc:f4:27:46:78:a7:
        41:00:ed:bc:9e:5c:e8:bc:fe:48:e2:77:3a:71:38:ea:b2:28:
        3b:a3:44:54:f2:c5:f7:b3:f8:87:f7:5f:5e:3b:17:ce:97:9c:
        d3:c6:52:26:1d:b0:98:4f:a3:ce:a8:17:d9:fb:da:22:6e:e5:
        ee:8d:04:df:2c:bb:9f:3d:89:af:7f:07:aa:c2:82:89:a0:b1:
        f0:42:a2:76:eb:d8:0c:9d:25:63:0f:46:f8:88:31:f8:a8:00:
        00:96:10:df:5e:4f:f3:f4:49:a6:e6:85:97:96:ca:41:fd:c1:
        55:26:e6:e8:df:ba:f6:63:01:85:36:3b:12:c9:e9:97:fc:fa:
        8d:52:19:4e:e1:2e:46:32:ca:f8:2b:47:c0:46:27:b4:78:75:
        be:64:df:6e
Certificate:
       Dat        Version: lu (0xlx)
        Serial Number: 1 (0x00000001)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: CN=OS Level_1 CA
        Validity
            Not Before: Feb 10 01:34:17 2006 GMT
            Not After : Feb 10 01:34:17 2007 GMT
        Subject: CN=OS Level_2 CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:a4:5f:c5:0f:cf:9b:05:b6:b2:31:16:bb:b1:c2:
                    be:35:58:a7:8b:ac:c2:1a:97:82:23:b0:2c:de:7c:
                    58:f0:97:ac:5d:7d:ef:8b:e2:82:1a:d4:d1:7e:38:
                    96:22:09:61:fd:73:36:d2:8c:3e:09:6b:e4:f1:f5:
                    d2:c7:2a:ed:4a:eb:f8:97:36:17:b3:e9:46:c9:f7:
                    6b:83:74:91:ff:cb:ed:5a:ad:d5:60:5c:2c:77:2a:
                    b2:62:23:0c:1c:af:4a:12:6e:30:54:7b:1b:96:f1:
                    30:40:23:39:f3:b6:09:a4:67:b1:65:d3:ef:05:32:
                    a7:a2:b8:7a:74:cc:18:9e:bc:e3:e4:89:f3:e5:36:
                    a0:c3:a9:e4:a1:27:49:08:a4:b2:3d:ae:76:11:69:
                    a0:32:c9:2e:43:94:4e:93:76:eb:5c:60:89:f2:a4:
                    c8:ec:1e:8d:fb:91:46:61:dc:c7:4b:5b:08:83:ef:
                    5c:e7:a1:2b:61:4c:87:58:2c:a0:1b:2f:34:21:82:
                    e7:ab:f0:62:d2:2c:52:7a:36:f8:c5:39:34:d4:27:
                    64:ae:47:83:d0:2d:a3:7c:0c:f2:5d:86:09:d1:3b:
                    3a:fd:0c:f6:93:a3:a3:c4:36:89:02:d0:41:bb:23:
                    14:03:9c:2e:05:54:bf:89:75:68:44:36:19:0a:2e:
                    14:b5
                Exponent: lu IÕ8~0xlx)
```

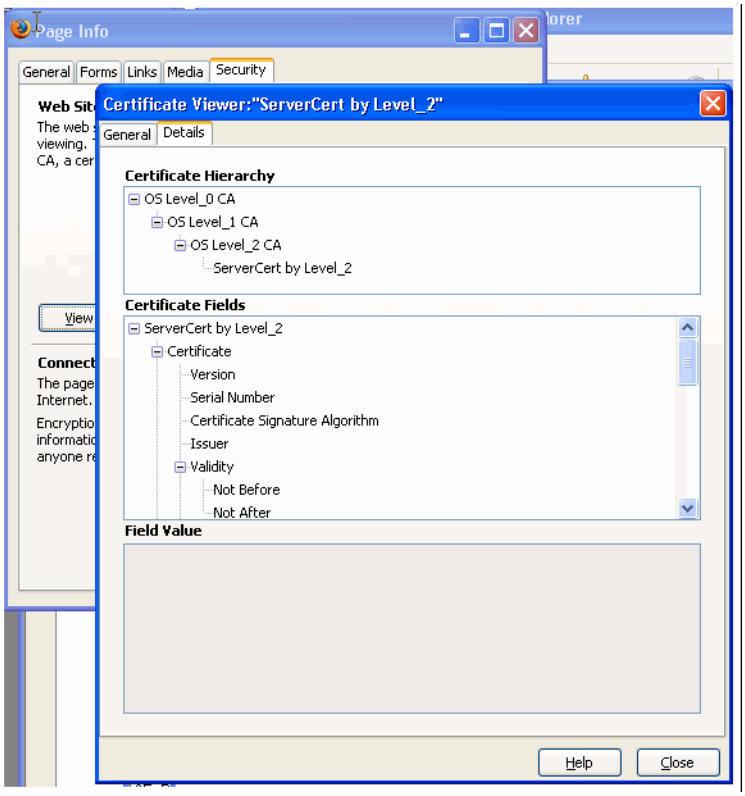
```
                Exponent: lu IÕ8~0xlx)
*s:
*sX509v3 Basic Constraints: critical
*sCA:TRUE
*sX509v3 Key Usage: critical
*sCertificate Sign, CRL Sign
*sNetscape Cert Type:
*sSSL CA, S/MIME CA, Object Signing CA
*sNetscape CA Revocation Url:
*s
*sX509v3 Subject Key Identifier:
*s
*sX509v3 Authority Key Identifier:
*skeyid:D6:D5:03:E1:B4:F0:0D:82:E9:AB:F0:4C:B2:FC:84:1B:82:18:8A:76
*sDirName:/CN=OS Level_0 CA
*sserial:01

*sAuthority Information Access:
*sCA Issuers - URI:http://s1.l47qa.com/l1/ca.crt

*sX509v3 CRL Distribution Points:
*sURI:http://s1.l47qa.com/l1crl-v2.crl

*sX509v3 Certificate Policies:
*sPolicy: 1.1.1.1.1
*sCPS:
*sUser Notice:
*sExplicit Text:

*sX509v3 Issuer Alternative Name:
*semail:root@s1.l47qa.com, URI:http://sq.l47qa.com
*sX509v3 Subject Alternative Name:
*s<EMPTY>

    Signature Algorithm: sha1WithRSAEncryption
        37:41:63:20:35:83:95:29:16:b1:ff:f4:7c:63:14:05:a2:f5:
        ef:df:6d:10:0c:26:4d:ed:60:15:b4:18:da:be:eb:e9:8a:15:
        82:f4:32:26:62:57:77:38:24:9a:f8:63:09:af:6c:7f:af:1c:
        de:ff:a4:50:7c:5e:38:b3:64:ff:03:a0:81:4d:3a:75:dd:e0:
        95:13:83:96:a1:dc:04:1f:4b:0b:59:b9:ec:8f:ea:f8:f4:fe:
        a0:58:92:14:3b:82:ec:3d:03:ab:2a:0c:96:74:b4:f0:4b:27:
        88:80:ee:9a:47:42:d5:c4:c8:ae:99:eb:c6:9e:65:66:30:13:
        e1:34:71:e0:b9:21:8b:4c:f4:7d:2f:08:4c:91:c3:ea:45:0e:
        25:cd:b1:b8:2f:a9:bd:53:82:da:de:48:49:36:d1:e5:0e:35:
        18:8a:59:7a:f3:21:c6:48:91:6b:17:70:b2:68:8a:b6:ae:15:
        68:7b:33:cd:cd:cd:de:71:a7:76:35:e5:cd:58:01:ae:44:ff:
        bc:50:7b:83:0b:8d:a2:83:1a:92:b1:b3:80:d8:e8:25:41:cb:
        78:c4:65:7f:af:de:f4:b6:47:e9:e8:11:56:0a:bd:73:71:cb:
        39:5b:70:a4:e3:77:3a:f3:44:f0:85:ba:e7:d1:65:dc:19:62:
        d5:44:11:27
Certificate:
       Dat        Version: lu (0xlx)
        Serial Number: 1 (0x00000001)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: CN=OS Level_0 CA
        Validity
            Not Before: Feb 10 01:30:07 2006 GMT
            Not After : Feb 10 01:30:07 2007 GMT
        Subject: CN=OS Level_1 CA
```

```
                RSA Public Key: (2048 bit)
                    Modulus (2048 bit):
                        00:a2:a9:48:46:79:dd:98:6b:9f:e9:77:b0:c7:eb:
                        37:ea:0a:7b:71:0d:5e:02:e6:d4:f7:1e:f2:9b:4f:
                        2d:f4:17:98:52:bc:13:5c:3b:83:84:f1:58:65:5b:
                        db:73:1b:38:96:c9:11:11:ca:6e:92:3c:80:9b:25:
                        3d:5a:78:15:93:00:a9:b8:82:9e:35:d3:13:1e:55:
                        9f:4f:87:03:d6:63:df:41:bd:51:85:5d:ef:b3:aa:
                        08:d9:80:43:9d:40:05:ae:10:f4:a1:0d:2c:32:b0:
                        d8:c5:50:59:65:01:a8:87:79:6e:f8:bf:6d:2a:90:
                        a0:06:f4:72:2a:26:6a:84:53:5a:0f:92:6e:07:1f:
                        d0:d6:6b:f9:2b:a3:3f:bb:e3:fe:bc:90:8d:fc:db:
                        6f:73:1b:41:40:78:b9:a3:8f:65:57:e9:11:74:a3:
                        55:3d:3b:c3:8e:fb:10:b2:03:0c:bc:cc:e4:d3:04:
                        9c:39:eb:b7:34:1b:a4:47:f4:88:2a:a2:23:61:d0:
                        f0:28:fe:ce:f5:b8:8f:a0:f0:de:1b:44:95:40:91:
                        55:c7:ee:14:45:b5:c7:48:28:8e:c0:4a:00:c1:23:
                        ac:9f:4e:00:b3:57:79:e6:12:d6:d7:e1:66:a2:62:
                        de:7f:13:b4:1f:17:1e:5a:22:ec:32:87:1a:87:a7:
                        73:cf
                    Exponent: lu IÕ8~0xlx)
*s:
*sX509v3 Basic Constraints: critical
*sCA:TRUE
*sX509v3 Key Usage: critical
*sCertificate Sign, CRL Sign
*sNetscape Cert Type:
*sSSL CA, S/MIME CA, Object Signing CA
*sNetscape CA Revocation Url:
*s
*sX509v3 Subject Key Identifier:
*s
*sX509v3 Authority Key Identifier:
*skeyid:9F:FF:BF:23:B9:CC:BE:3B:BA:97:94:60:01:60:FB:F9:EF:E8:54:A3
*sDirName:/CN=OS Level_0 CA
*sserial:00

*sAuthority Information Access:
*sCA Issuers - URI:http://s1.l47qa.com/ca.crt

*sX509v3 CRL Distribution Points:
*sURI:http://s1.l47qa.com/crl-v2.crl

*sX509v3 Certificate Policies:
*sPolicy: 1.1.1.1.1
*sCPS:
*sUser Notice:
*sExplicit Text:

*sX509v3 Issuer Alternative Name:
*semail:root@s1.l47qa.com, URI:http://s1.l47qa.com
*sX509v3 Subject Alternative Name:
*s<EMPTY>
    Signature Algorithm: sha1WithRSAEncryption
        95:2a:1a:b2:c0:07:54:ed:72:25:88:70:8e:b0:af:89:43:7b:
            a9:fe:19:9d:0e:c1:b0:f9:73:ec:66:db:38:30:39:91:92:71:
```

```
d3:c2:64:4d:24:41:5a:2c:17:3d:34:27:8b:0c:25:60:6b:3a:
86:f6:54:fc:8d:31:08:3b:dd:4c:cb:46:fb:47:a3:e4:23:3d:
82:33:84:d2:fb:81:05:61:95:09:98:a4:25:f0:55:eb:80:0c:
32:69:48:cf:41:7c:36:2d:d7:c0:02:79:a1:7b:4d:28:4c:84:
64:68:3c:8a:af:28:5f:f6:78:1e:31:d4:5a:2c:60:20:12:99:
5c:e3:df:59:01:79:7c:20:c8:f5:ab:75:e6:ab:db:de:2a:e7:
be:4d:a1:9d:d5:5a:7c:9a:22:14:ca:7b:31:9d:48:d8:62:3a:
ab:97:15:6b:4f:13:3e:35:c0:fb:82:57:20:e7:08:03:33:28:
19:20:16:24:28:98:d4:f7:cf:0b:4b:0c:7e:6a:88:54:b0:06:
2e:df:b3:6e:ea:8e:4a:a0:60:78:73:40:a7:75:80:ef:96:cb:
f1:03:96:83:cf:1a:38:a7:33:82:d5:2e:e7:51:93:06:59:b5:
95:16:a4:34:d4:63:e7:9f:6e:7b:aa:30:13:ed:3e:47:a1:b9:
f8:56:d6:11
```

Figure 14 shows the certificate hierarchy.

FIGURE 14 Certificate Hierarchy



The root CA certificate "OS level 0 CA" was not included in the chain because it was already trusted by the client, and the client accepted the chain, as shown in Figure 14.

Common Mistakes

The following mistakes often occur when dealing with server certificates and CA chains:

- Symptom: The certificate chain is not properly uploaded.

Solution: To verify that the certificate chain is properly uploaded on ServerIronADX, connect to the BP console and enter the **show ssl certificate** <cert-name> command. Make sure that all of the intermediate CA certificates are included.

- Symptom: The wrong format was specified when uploading the certificate. For example, the certificate was obtained in DER format but uploaded in PEM format.

  Solution: Check the certificate from the BP console to make sure that it is readable.

- Symptom: The certificate is chained but "enable-certificate-chaining" command is not enabled under ssl profile

  Solution: If chained certificates are used then "enable-certificate-chaining" command is required under ssl profile definition. The chaining is disabled by default to improve system performance.

- Symptom: Certificate is signed by Verisign, but it is shown as expired when the client tries to connect to the ServerIron. This usually occurs if the client browser has an expired copy of the intermediate CA certificate.

  Solution: Refer to the following URL from Verisign Web site. Download the current copy of intermediate CA certificate and either replace the expired copy on client browser with the new copy or append the intermediate CA certificate to the server certificate on the ServerIron.

  http://knowledge.verisign.com/search/solution.jsp?id=vs5781

  Perform steps 1 and 2 in the following sections to append intermediate CA certificates to server certificates.

Step 1: Import Server Certificate and Intermediate CA Certificates

To chain SSL certificates, follow these steps:

1. Import the server certificate using the following command:

   ```
   scp -1 /home/rr/server.crt rr@30.30.1.1:sslcert:chain2cert:pem
   ```

2. Import the intermediate CA certificate using the following command:

   ```
   scp -1 /home/rr/inter.crt rr@30.30.1.1:sslcert:chain2cert:pem
   ```

NOTE
In this example, **server.crt** is issued by intermediate CA. The **inter.crt** is the intermediate CA certificate.

NOTE
The order is important. The server certificate should be imported before the intermediate CA certificate.

NOTE
The same file name should be used (chain2cert in this example) when importing both the server and intermediate CA certificate.

Step 2: Enable Certificate Chain

By default, for CA signed certificates, the ServerIronADX does not send the entire certificate chain when presenting the certificate to the client.

To enable the ServerIronADX to send the entire certificate chain configure the enable-certificate-chaining command within an SSL profile as described in "Enabling a certificate chain" on page 169.

Support for SSL renegotiation

Some SSL application clients use renegotiation as a way within SSL protocols to change cipher specifications and redo the handshake. It has been found however that unsecure renegotiation is susceptible to Man-in-the-Middle attack.

Although ServerIron ADX does not support renegotiation and is therefor not susceptible to these attacks, it doesn't handle renegotiation requests from the client properly in some cases which causes some web browsers to report a security flaw with ServerIron ADX which is a false alarm.

With this feature enabled as shown, a ServerIron ADX responds to renegotiation requests which stops the browser from sending false alarms.

```
ServerIronADX(config)# server respond-with-renegotiation-info
```

Syntax: {no} ssl server respond-with-renegotiation-info

With this command enabled, a ServerIron ADX will look for renegotitation-related heaters in SSL packets and respond accordingly.

Where this command is not enabled, a ServerIron ADX ignores all renegotiation-related headers.

NOTE
While a ServerIron ADX with this command enabled will respond to renegotiation requests, ServerIron ADX does not currently support renegotiation.

Basic SSL profile configuration

All SSL configuration parameters are configured in the configuration level under the specific SSL profile. An SSL profile is created using the **ssl profile** command at the General configuration level as shown.

```
ServerIronADX(config)# ssl profile profile1
ServerIronADX(config-ssl-profile-profile1)#
```

Syntax: ssl profile *<profile-name>*

The <profile-name> variable is an ASCII string that specifies the name of the SSL profile being defined.

At a minimum the following parameters need to be configured for an SSL profile:

- The RSA key-pair for the SSL connection
- The cipher suite for the SSL connection
- The digital certificate for the SSL connection (specified or self-signed)

Specifying a keypair file

Each SSL profile must be associated with an RSA key-pair file that was previously defined using the **genrsa** command. The following example uses the **keypair-file** command to associate the key pair file named "rsakey" with the "profile1" SSL profile.

```
ServerIronADX(config)# ssl profile profile1
ServerIronADX(config-ssl-profile-profile1)# keypair-file rsakey
```

Syntax: keypair-file <*keypair-file-name*>

The <keypair-file-name> variable is an ASCII string a keypair file that was generated using the **genrsa** command.

Specifying a cipher suite

By specifying cipher suites under an SSL profile, you can control the security strength of the SSL handshakes. The ServerIronADX can accept a new SSL handshake from the client only if the list of cipher suites presented by the client includes a cipher suite configured under the SSL profile.

The following example specifies that all cipher suites are configured under the "profile1" SSL profile.

```
ServerIronADX(config)# ssl profile profile1
ServerIronADX(config-ssl-profile-profile1)# cipher-suite all-cipher-profiles
```

Syntax: cipher-suite rsa-export-with-des40-cbc-sha | rsa-export-with-rc4-40-md5 | rsa-with-3des-ede-cbc-sha | rsa-with-aes-128-sha | rsa-with-aes-256-sha | rsa-with-des-cbc-sha | rsa-with-rc4-128-md5 | rsa-with-rc4-128-sha | all-cipher-suites

Use the **rsa-export-with-des40-cbc-sha** parameter to specify that cipher suite.

Use the **rsa-export-with-rc4-40-md5** parameter to specify that cipher suite.

Use the **rsa-with-3des-ede-cbc-sha** parameter to specify that cipher suite.

Use the **rsa-with-aes-128-sha** parameter to specify that cipher suite.

Use the **rsa-with-aes-256-sha** parameter to specify that cipher suite.

Use the **rsa-with-des-cbc-sha** parameter to specify that cipher suite.

Use the **rsa-with-rc4-128-md5** parameter to specify that cipher suite.

Use the **rsa-with-rc4-128-sha** parameter to specify that cipher suite.

Use the **all-cipher-suites** parameter to specify all cipher suites including the other parameters used in this command.

NOTE
The export cipher suites work only if the asymmetric key pair strength is less than or equal to 512 bits. This is consistent with the export rules. If the RSA key pair strength is greater than 512 bits, then SSL handshake requests that contain export cipher suites do not work.

Configuring Multiple Cipher Suites

Among the cipher suite options, is one that specifies all cipher suites. You can also specify more than one cipher inside an SSL profile without specifying all options. This is shown in the following example.

To configure this feature, use commands such as the following:

```
ServerIronADX(config)#ssl profile sp1
ServerIronADX(config-ssl-profile-sp1)# cipher-suite rsa-with-aes-128-sha
ServerIronADX(config-ssl-profile-sp1)# cipher-suite rsa-with-rc4-128-md5
ServerIronADX(config-ssl-profile-sp1)# cipher-suite rsa-with-rc4-128-sha
```

Specifying a certificate file

Each SSL profile must be associated with a certificate file that was either imported or self generated as described in "Chained certificates" on page 139. The following example uses the **certificate-file** command to associate the certificate file named "certfile1" with the "profile1" SSL profile.

```
ServerIronADX(config)# ssl profile profile1
ServerIronADX(config-ssl-profile-profile1)# certificate-file certfile1
```

Syntax: certificate-file <*certificate-file-name*>

The <certificate-file-name> variable is an ASCII string that specifies a certificate file that either self generated on the ServerIronADX using the **ssl gencert** command or imported into the ServerIronADX as described in "Chained certificates" on page 139.

Advanced SSL profile configuration

This section describes the following advanced SSL configuration options:

- Client authentication
- Enabling Session caching
- Enabling SSLv2
- Enabling close notify
- Disabling Certificate verification

All SSL configuration parameters are configured in the configuration level under the specific SSL profile. An SSL profile is created using the **ssl profile** command at the General configuration level as shown in "Basic SSL profile configuration".

Configuring client authentication

The following features can be configured for certificate management:

- Enabling certificate verification
- Configuring a CA certificate file
- Creating a certificate revocation list
- Allowing self signed certificates

Enabling certificate verification

The ServerIronADX can be optionally configured to enforce client certificate verification. When client certificate verification is configured, the ServerIronADX requires all clients to present their signed certificates. The certificates are compared against trusted CAs and a connection is allowed or denied.

You can enable client certificate verification on a per-ssl-handshake or per-connection basis in one of two modes:

- Request mode
- Require mode

In request mode, a client-certificate is requested. The connection is allowed if the client presents a valid certificate, or if a certificate is not presented at all. The connection is denied if a client presents an invalid, revoked, or expired certificate.

In require mode, a client-certificate is always required.

Client-authentication can be used in the following four combinations:

- Per-connection request
- Per-connection require
- Per-ssl-handshake request
- Per-ssl-handshake require

Syntax: verify-client-cert <*per-ssl-handshake/per-ssl-connection*> <*request/require*>

- per-ssl-handshake - Requests a client certificate for every new SSL handshake.
- per-connection - Requests a client certificate for every new SSL connection.

The difference between the two modes is apparent if SSL session caching is enabled. When this is the case, multiple SSL connections share the same SSL session, without performing a full SSL handshake for each connection.

Client certificate verification in SSL Proxy Mode

SSL Proxy mode has two traffic segments: from the client to the ServerIronADX and from the ServerIronADX to the server.

In the first segment, the ServerIronADX acts a server to a browser-based client. In the second segment, ServerIronADX acts as a client to the real server.

In some cases the real server is configured so that only clients with valid certificates can connect to it. Because the ServerIronADX is also a client, it must have a valid client certificate to connect to the real server. A client certificate can be obtained from a CA, and uploaded to the ServerIronADX. Once uploaded, the client certificate should be configured in the server ssl profile using the following commands:

- keypair-file - To configure client-certificate key
- certificate file - To configure client-certificate

Client certificate verification in the second traffic segment (from the ServerIronADX to the server) can also be enabled. In this configuration, the real server allows a connection only from the ServerIronADX. No other device is allowed. To connect to the real server, the ServerIronADX must present a client certificate issued by a CA and trusted by the server.

To successfully complete this process, the ServerIronADX requires the following items:

- A certificate issued by a CA that is trusted by the server
- A key-pair for the certificate

The certificate and the key can be obtained from the CA in either PKCS or PEM format. For client-authentication to work, these items must be uploaded to the ServerIronADX and then added to the server profile.

For example, if you use si_client_cert.pem as the certificate and si_client_key.pem as the key for the client certificate, you can add them to the profile using the following commands:

```
ServerIronADX(config)# ssl profile serverProfile
ServerIronADX(config-ssl-profile-serverProfile)# keypair-file si_client_key.pem
ServerIronADX(config-ssl-profile-serverProfile)# certificate-file
si_client_cert.pem
```

Configuring a CA certificate file

If you have enabled client certificate verification, you must configure a CA certificate under the SSL profile. CA certificates are used by the ServerIronADX to verify the validity of certificates presented by incoming clients.

CA certificates are typically imported from outside using SCP, in PEM format and are stored in the flash memory, just like regular certificate files.

Up to four CA certificate files can be specified under each SSL profile. Each CA certificate file can contain multiple CA certificates (although to keep configurations simple, We recommend that different CA certificates be stored in different files).

You can include up to 32 DN names for all root or intermediate CA certificates. This allows clients to select appropriate CA and intermediate CA certificates for communication with a ServerIronADX.

Unlike regular certificates, there is no need to load the corresponding key pair into the profile before configuring a CA certificate since the CA certificate belongs to the Certificate Signing Authority, meaning the key pair is private and not be publicly available. The following example specifies the CA certificate file named "certfile1" for SSL profile "profile1".

```
ServerIronADX(config)# ssl profile profile1
ServerIronADX(config-ssl-profile-profile1)# ca-cert-file certfile1
```

Syntax: ca-cert-file *<ca-certificate-filename>*

The *<ca-certificate-filename>* variable specifies the name of the certificate file where a CA certificate is stored.

NOTE
You can optionally disable certificate verification as described in "Disabling certificate verification" on page 171.

Creating a certificate revocation list

Certificate revocation lists contain the list of certificates that have been revoked by a CA. A certificate can be revoked by a CA for many reasons. A common reason is that the key pair that corresponds to the issued certificate has been compromised.

Certificate revocation lists are typically maintained on the CA Web site and may be downloaded using HTTP. The format of the list is usually DER or PEM.

The ServerIronADX supports configuration of up to ten CRL records. For each CRL record, the size is up to 255K.

Syntax: ssl crl-record <*local-name*> <*url*> **der | pem** <*refresh-interval-in-hours*>

The <local-name> variable specifies a name for the CRL entry. The value of this entry is an ASCII string.

The <url> variable specifies the location where the CRL is located. This value can be either an IP address or a domain name.

The **pem** parameter directs the CRL to be downloaded in the PEM format.

The **der** parameter directs the CRL to be downloaded in the DER format.

The <refresh-interval-in-hours> variable specifies the number of hours to wait before updating the CRL.

NOTE
Limiting the maximum number of connections from all client-ip's is supported only via the **max-conn default <num>** command. The **max-conn 0.0.0.0/0 <num>** command is no longer supported.

NOTE
To avoid "man-in-the-middle" attacks, where the CRL may be compromised while on the network, CRLs are digitally signed by the issuing CAs. For this reason, it is essential that the certificate of the CA that issues the CRL is present on the ServerIronADX when a client certificate is being checked for revocation.

Allowing Self Signed Certificates

By default, the a ServerIronADX does not accept certificates that have been issued by a CA that is not trusted. A ServerIronADX only accepts certificates which have been signed by a CA that is configured under the SSL profile. For testing purposes, customers may want to use self-signed certificates (generated using the Open SSL utilities or by the ServerIron cert gen utility) on the SSL client.

The following example configures a ServerIronADX to accept self signed certificates.

```
ServerIronADX(config)# ssl profile profile1
ServerIronADX(config-ssl-profile-profile1)#allow-self-signed-cert
```

Syntax: [no] allow-self-signed-cert

Enabling a certificate chain

By default, for CA signed certificates, the ServerIronADX does not send the entire certificate chain when presenting the certificate to the client.

To enable the ServerIronADX to send the entire certificate chain (including the root CA certificate and any intermediate CA certificates), enter the following commands in the SSL profile configuration mode:

```
ServerIronADX(config)#ssl profile profile1
```

Syntax: ssl profile <*profile-name*>

```
ServerIronADX(config-ssl-profile-ssl-profile1)# enable-certificate-chaining
```

Syntax: enable-certificate-chaining

NOTE
All intermediate CA certificates need to be uploaded to the ServerIronADX.

Configuring certificate chain depth

You can configure certificate chain depth up to which certificate verification can be done by a ServerIronADX. The default value is 4 and it can be configured up to 10 as shown in the following.

```
ServerIronADX(config)#ssl profile profile1
ServerIronADX(config-ssl-profile-ssl-profile1)# verify-cert-depth 10
```

Syntax: [no] verify-cert-depth <*chain-depth*>

The <*chain-depth*> variable specifies the maximum certificate chain depth verified. The accepted values are 4 - 10. The default value is 4.

Enabling session caching

Session caching or session reuse is a mode of operation in SSL where multiple SSL connections can share the same SSL session. A complete SSL handshake is done only for the first connection. All subsequent connections use the parameters negotiated in the first connection, for as long as the SSL session is cached.

By default, session caching is turned off on the ServerIronADX.

The following example enables session caching for the SSL client in the SSL profile "profile1".

```
ServerIronADX(config)# ssl profile profile1
ServerIronADX(config-ssl-profile-profile1)# session-cache on
```

Syntax: [no] session-cache { on | off }

The **on** parameter enables session caching for the SSL client.

The **off** parameter disables session caching. This is the default state.

NOTE
Please note that SSL session caching will not work with the **server source-port-hash** command because that command will redirect traffic (from the same client IP) with different TCP source ports to different BPs.

Configuring session cache size

You can specify the maximum number of session-cache entries per profile, as shown in the following example:

```
ServerIronADX(config-ssl-profile-ssl1)# session-cache max-entries 512
```

Syntax: [no] session-cache max-entries <*num-max-entries*>

The <num-max-entries> can have a value between 512 and 8192.

The default value is 1024.

Configuring a session cache timeout

By default, SSL sessions are held in the cache for 30 seconds. You can change the time period a session is in cache, as shown in the following.

```
ServerIronADX(config)# ssl profile profile1
ServerIronADX(config-ssl-profile-profile1)# session-cache-timeout
```

Syntax: [no] session-cache-timeout <*timeout-in-seconds*>

The <timeout-in-seconds> variable can be set to a value between 20 and 86400 seconds. The default value is 30 seconds.

Enabling SSL Version 2

By default, the ServerIronADX supports SSL version 3. You can enable SSL version 2 as shown in the following example.

To do this, enter the following command under the SSL profile:

```
ServerIronADX(config)# ssl profile profile1
ServerIronADX(config-ssl-profile-profile1)# enable-ssl-v2
```

Syntax: [no] enable-ssl-v2

SSLv2 is disabled by default.

Enabling close notify

You can configure a ServerIronADX to send an alert before closing an SSL session as shown in the following.

```
ServerIronADX(config)# ssl profile profile1
ServerIronADX(config-ssl-profile-profile1)# enable-close-notify
```

Syntax: [no] enable-close-notify

When this command is configured, the ServerIronADX will send an alert before closing an SSL session. By default, a ServerIronADX does not send a close notify alert before closing an SSL session.

Disabling certificate verification

You can configure an ServerIron ADX to disable certificate verification as shown in the following:

```
ServerIronADX(config)# ssl profile profile1
ServerIronADX(config-ssl-profile-profile1)# disable-certificate-checking
```

Syntax: [no] disable-certificate-checking

This command only applies to SSL proxy mode. When a ServerIron ADX is in SSL proxy mode, it acts as a client for the backend server.

By default, if the server sends a certificate with the wrong information, the ServerIron ADX will reject it. If this command is configured, the ServerIron ADX will accept an invalid certificate.

Enabling a ServerIron ADX SSL to respond with renegotiation headers

Some SSL application clients use renegotiation as a way within SSL protocols to change cipher specifications and redo the handshake. It has been reported that unsecure renegotiation is susceptible to Man-in-the-Middle attack. ServerIron ADX does not support renegotiation. This means that ServerIron ADX is not susceptible to these attacks.

A problem occurs however where some Web browsers using OpenSSL send renegotiation related headers and expect a response. If a ServerIron ADX does not respond with an appropriate header for renegotiation, these web browers miss-intreprete the ServerIron ADX to be vulnerable to renegotiation attacks.

With release 12.4.00, an option has been added to configure a ServerIron ADX to respond with renegotiation headers that tell the browers that the ServerIron ADX handles the renegotaiton message correctly and stops them from sending the false message that the ServerIron ADX is vulnerable to renegotiation attacks.

Configuring this command as shown in the following does not enable renegotiation on the ServerIron ADX but prevents the problem with false reporting.

```
ServerIronADX# server ssl respond-with-renegotiation-info
```

Syntax: [no] server ssl respond-with-renegotiation-info

NOTE
The ServerIron ADX will still not support renegotiation. If the client attempts to renegotiate, the ServerIron ADX will immediately terminate the handshake with the "NO_Renegotiation" handshake message. However since the ServerIron ADX is now responding to the renegotiation headers, OpenSSL clients that did not have any problem with ServerIron ADX NOT supporting renegotiation might now be mislead to believe that ServerIron ADX has started supporting renegotiation. If this occurs you may need to turn off this feature using the **no** option.

Configuring Real and Virtual Servers for SSL Termination and Proxy Mode

When configuring a ServerIron ADX for SSL Termination and Proxy mode, the Real and Virtual Servers need to be configured to support these features. the following sections describe the procedures and commands required. For a description of SSL Termination Mode, see "SSL Termination Mode" on page 137. For a description of SSL Proxy Mode, see "SSL Proxy Mode" on page 138. For a detailed example of how to configure the examples shown in those sections, see "Configuration Examples for SSL Termination and Proxy Modes" on page 176.

NOTE
SSL Termination and Proxy mode can be configured for setups where an IPv4 real server is bound to an IPv4 virtual server or where an IPv6 real server is bound to an IPv6 virtual server. They are not supported for setups that use IPv4 and IPv6 together in the same configuration.

Configuring Real and Virtual Servers for SSL Termination Mode

Real and Virtual Server configuration is described in detail in the *Brocade ServerIron ADX Server Load Balancing Guide*. When configuring a Real or Virtual Server for SSL Termination Mode, you need to do the following:

- Configure a Real Server with an HTTP port
- Configure a Virtual Server with an SSL port
- Enable SSL termination and specify an SSL profile on the SSL port of the Virtual Server
- Bind SSL on the Virtual Server to an HTTP port on a Real Server

For IPv4 Real Server to IPv4 Virtual Server

In the example below an IPv4 Real Server and a IPv4 Virtual Server are configured for SSL Termination mode with the following details:

- An HTTP port is defined on the Real Server: "rs1"
- An SSL port is defined on the Virtual Server: "vip1".
- SSL Termination is enabled and the SSL profile "myprofile" is specified on the Virtual Server: "vip1".
- A bind is configured between SSL on Virtual Server: "vip1" and HTTP on Real Server: "rs1".

```
ServerIronADX(config)# server real rs1 10.1.1.1
ServerIronADX(config-rs-rs1)# port http
ServerIronADX(config-rs-rs1)# exit
ServerIronADX(config)# server virtual-name-or-ip vip1
ServerIronADX(config-vs-vip1)# port ssl
ServerIronADX(config-vs-vip1)# port ssl ssl-terminate myprofile
ServerIronADX(config-vs-vip1)# bind ssl rs1 http
```

For IPv6Real Server to IPv6 Virtual Server

In the example below an IPv6 Real Server and a IPv6 Virtual Server are configured for SSL Termination mode with the following details:

- An HTTP port is defined on the Real Server: "rs2"
- An SSL port is defined on the Virtual Server: "vip2".
- SSL Termination is enabled and the SSL profile "ipv6_profile" is specified on the Virtual Server: "vip2".
- A bind is configured between SSL on Virtual Server: "vip2" and HTTP on Real Server: "rs2".

```
ServerIronADX(config)# server real rs2 2000::1
ServerIronADX(config-rs-rs2)# port http
ServerIronADX(config-rs-rs2)# exit
ServerIronADX(config)# server virtual-name-or-ip vip2
ServerIronADX(config-vs-vip2)# port ssl
ServerIronADX(config-vs-vip2)# port ssl ssl-terminate ipv6_profile
ServerIronADX(config-vs-vip2)# bind ssl rs2 http
```

Syntax: [no] port ssl ssl-terminate *<ssl-profile-name>*

The *<ssl-profile-name>* variable specifies the name of the SSL profile that you want to bind to the SSL port, termination mode configuration.

Configuring Real and Virtual Servers for SSL Proxy Mode

Real and Virtual Server configuration is described in detail in the *ServerIron ADX Server Load Balancing Guide*. When configuring a Real or Virtual Server for SSL Proxy Mode, you need to do the following:

- Configure a Real Server with an SSL port
- Configure a Virtual Server with an SSL port
- Enable SSL Proxy and specify an SSL client profile and an SSL server profile on the SSL port of the Virtual Server
- Bind SSL on the Virtual Server to an SSL port on a Real Server

For IPv4 Real Server to IPv4 Virtual Server

In the example below an IPv4 Real Server and an IPv4 Virtual Server are configured for SSL Proxy mode with the following details:

- An SSL port is defined on the Real Server: "rs3"
- An SSL port is defined on the Virtual Server: "vip3".
- SSL Proxy is configured and the SSL client profile "IPv4clientprofile" and SSL server profile "IPv4serverprofile" are specified on the Virtual Server: "vip3".
- A bind is configured between SSL on Virtual Server: "vip3" and SSL on the Real Server: "rs3".

```
ServerIronADX(config)# server real rs3 10.1.1.1
ServerIronADX(config-rs-rs3)# port ssl
ServerIronADX(config-rs-rs3)# exit
ServerIronADX(config)# server virtual-name-or-ip vip3
ServerIronADX(config-vs-vip3)# port ssl
ServerIronADX(config-vs-vip3)# port ssl ssl-proxy IPv4clientprofile
IPv4serverprofile
ServerIronADX(config-vs-vip3)# bind ssl rs3 ssl
```

For IPv6 Real Server to IPv6 Virtual Server

In the example below an IPv6 Real Server and an IPv6 Virtual Server are configured for SSL Proxy mode with the following details:

- An SSL port is defined on the Real Server: "rs4"
- An SSL port is defined on the Virtual Server: "vip4".
- SSL Proxy is configured and the SSL client profile "IPv6clientprofile" and SSL server profile "IPv6serverprofile" are specified on the Virtual Server: "vip4".
- A bind is configured between SSL on Virtual Server: "vip4" and SSL on the Real Server: "rs4".

```
ServerIronADX(config)# server real rs4 2000::2
ServerIronADX(config-rs-rs4)# port ssl
ServerIronADX(config-rs-rs4)# exit
ServerIronADX(config)# server virtual-name-or-ip vip4
ServerIronADX(config-vs-vip4)# port ssl
ServerIronADX(config-vs-vip4)# port ssl ssl-proxy IPv6clientprofile
IPv6serverprofile
ServerIronADX(config-vs-vip4)# bind ssl rs4 ssl
```

Syntax: [no] port ssl ssl-proxy <*ssl-profile-name-1*> <*ssl-profile-name-2*>

The <*ssl-profile-name-1*> and <*ssl-profile-name-2*> variables specify the name of the SSL profiles that you want to bind to the SSL port, proxy mode configuration. The first profile is used for the client to ServerIron ADX side and the second profile is used for the ServerIron ADX to the Real Server side.

NOTE
The ServerIron ADX SSL proxy mode does not support session reuse that takes place on the ServerIron ADX to the Real Server side.

Configuration Examples for SSL Termination and Proxy Modes

This section describes the procedures required to perform the configurations described in "SSL Termination Mode" on page 137 and "SSL Proxy Mode" on page 138. As shown in the examples there, SSL Termination mode provides for an SSL connection between clients to the ServerIron ADX. When configuring SSL Proxy Mode a configuration is created between the ServerIron ADX and the server. In this case, the ServerIron ADX is configured as a client to the server.

Configuring SSL Termination Mode

In this mode, for enabling VRRPE for VIP address, it is necessary to use a different **source-nat-ip** for ssl traffic.

For performing this function, use the following syntax:

Syntax: server source-nat-ip *<ip> <mask> <gateway>* **port-range** *<range>*

To configure SSL in the termination mode, perform the following tasks in sequence:

1. Generate or obtain an RSA key pair and copy it to the ServerIron ADX
2. Obtain a digital certificate and copy it to the ServerIron ADX
3. Create an SSL profile as described in "Allowing Self Signed Certificates" on page 169
4. Within the SSL profile specify a keypair file as described in "Specifying a keypair file" on page 165.
5. Within the SSL profile specify a digital certificate file as described in "Specifying a certificate file" on page 166.
6. Within the SSL profile select a Cipher Suite as described in "Specifying a cipher suite" on page 165. This is optional.
7. Configure Real and Virtual Servers as described in "Configuring Real and Virtual Servers for SSL Termination Mode" on page 173

Example

Generate an RSA key pair

```
ServerIronADX# ssl genrsa rsakey-file 1024 mypassword
```

Generate a Self-signed Digital Certificate

```
ServerIronADX# ssl gencert certkey rsakey-file signkey rsakey-file mypassword
mycert
You are about to be asked to enter information that will be incorporated into
your certificate request. What you are about to enter is what is called a
Distinguished Name or a DN.
Country name (2 letter code) [US] US
State or province (full name) [California] California
Locality name (city) [city]San Jose
Organization name (Company name) [Brocade] Brocade
Organizational unit name (department) [Web administration] Web Administration
Common name (your domain name) [www.brocade.com] www.brocade.com
Email address [webadmin@brocade.com] webadmin@brocade.com
transfer_ssl_object_buf_to_bp : The object buffer length is 492
transfer_ssl_object_buf_to_bp: The message length is 622
```

Create SSL profile with required settings

```
ServerIronADX(config)# ssl profile myprofile
ServerIronADX(config-ssl-profile-myprofile)# keypair-file rsakey-file
ServerIronADX(config-ssl-profile-myprofile)# certificate-file mycert
ServerIronADX(config-ssl-profile-myprofile)# cipher-suite all
ServerIronADX(config-ssl-profile-myprofile)# exit
```

Define HTTP ports on real servers

```
ServerIronADX(config)# server real rs1 10.1.1.1
ServerIronADX(config-rs-rs1)# port http
ServerIronADX(config-rs-rs1)# exit
ServerIronADX(config)# server real rs2 10.1.1.2
ServerIronADX(config-rs-rs2)# port http
ServerIronADX(config-rs-rs2)# exit
```

Within virtual server: Define SSL port, specify server profile and enable SSL terminate

```
ServerIronADX(config)# server virtual-name-or-ip vip1 10.1.1.7
ServerIronADX(config-vs-vip1)# port ssl
ServerIronADX(config-vs-vip1)# port ssl ssl-terminate myprofile
```

Bind SSL in virtual server to real server HTTP ports

```
ServerIronADX(config-vs-vip1)# bind ssl rs1 http rs2 http
```

Configuring SSL Proxy Mode

The ServerIron ADX acts as a client to the real server. The real server presents a certificate, but the certificate needs to be verified by the ServerIron ADX. Because the ServerIron ADX needs the CA certificate from the issuing authority to verify the certificate from the real server, the CA certificate must be uploaded to the ServerIron ADX before it can be used.

To configure SSL in proxy mode, perform the following tasks in sequence:

1. Upload the CA certificate to the ServerIron ADX as described in "Transferring a Keypair File and a Certificate File" on page 149.

   NOTE
   If the server is using a self-signed certificate, the **allow-self-signed certificate** command must be configured within the profile.

2. Create a Client Side SSL Profile.
3. Associate an RSA key pair and certificate with the Client Side SSL Profile.
4. Within the Client Side SSL profile select a Cipher Suite as described in "Specifying a cipher suite" on page 165. This is optional.
5. Create a Server Side SSL Profile
6. In the Server Side profile specify the name of the certificate to be associated with the SSL Server Side profile.
7. Configure Real and Virtual Servers as described in "Configuring Real and Virtual Servers for SSL Proxy Mode" on page 174

Example

Create Client Side SSL profile with required settings

```
ServerIronADX(config)# ssl profile clientprofile
ServerIronADX(config-ssl-profile-clientprofile)# keypair-file rsakey-file
ServerIronADX(config-ssl-profile-clientprofile)# certificate-file mycert
ServerIronADX(config-ssl-profile-clientprofile)# cipher-suite all
ServerIronADX(config-ssl-profile-clientprofile)# exit
```

Create server side SSL profile with required settings

```
ServerIronADX(config)# ssl profile serverprofile
ServerIronADX(config-ssl-profile-serverprofile)# ca-cert-file ca.cert
ServerIronADX(config-ssl-profile-clientprofile)# cipher-suite all
```

Define SSL ports on real servers

```
ServerIronADX(config)# server real rs1 10.1.1.1
ServerIronADX(config-rs-rs1)# port ssl
```

Within virtual server: Define SSL port, specify server profile and enable SSL proxy

```
ServerIronADX(config)# server virtual-name-or-ip vip3 10.1.1.3
ServerIronADX(config-vs-vip3)# port ssl
ServerIronADX(config-vs-vip3)# port ssl ssl-proxy clientprofile serverprofile
```

Bind SSL in virtual server to real server SSL ports

```
ServerIronADX(config-vs-vip3)# bind ssl rs1 ssl
```

TCP configuration issues with SSL Terminate and SSL Proxy

When SSL terminate or SSL proxy are enabled, the ServerIron ADX uses TCP full stack. In such case, the Nagle Algorithm and delayed ACK mechanism are ON by default. There are instances where both of these features should be disabled.

For example, a customer may be experiencing slow response time because the ServerIron ADX is sending one packet at a time, and waiting for an ACK from the server before sending the next packet. The server is sending ACKs with a delay of 200 ms, causing a delay of 200 ms between every successive packet. This results in extremely poor performance. Packet traces taken from the client and server sides explain this situation in detail, as shown in the following figures.

Figure 15 shows the client ptrace information. Figure 16 shows the server ptrace information.

FIGURE 15 Client Capture

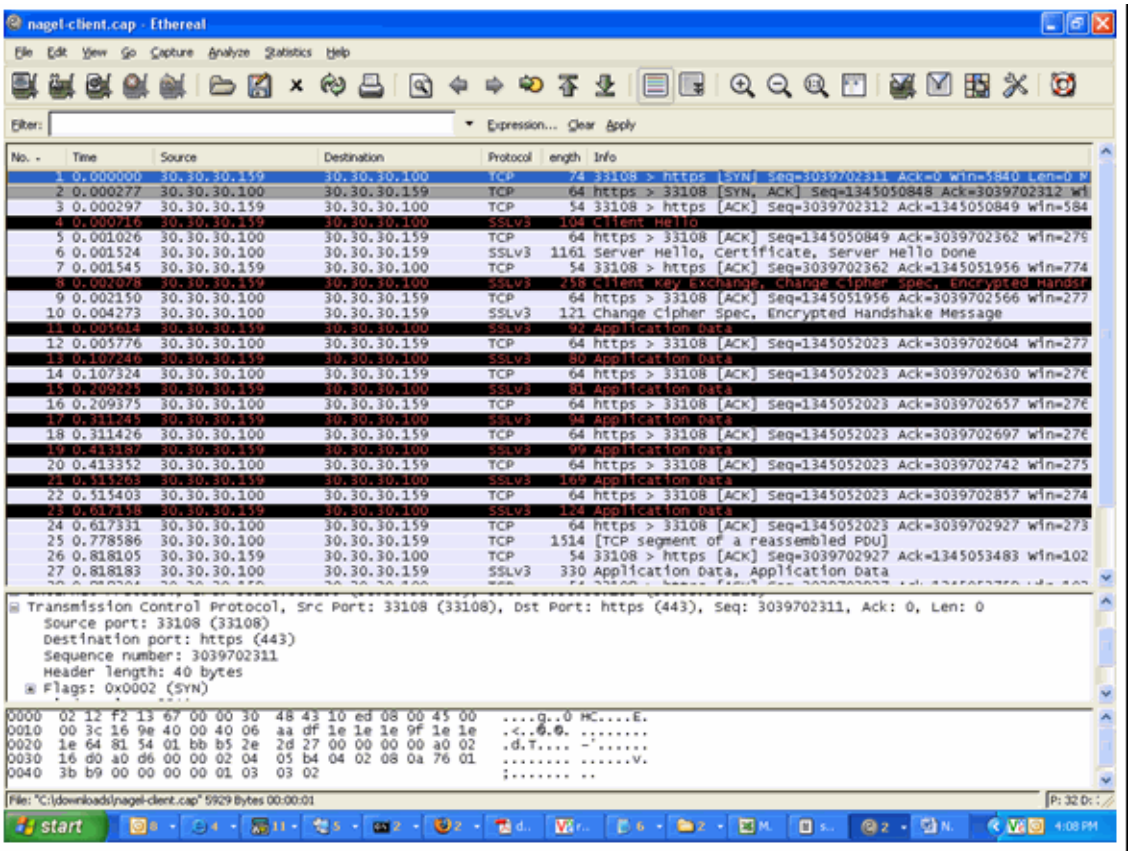
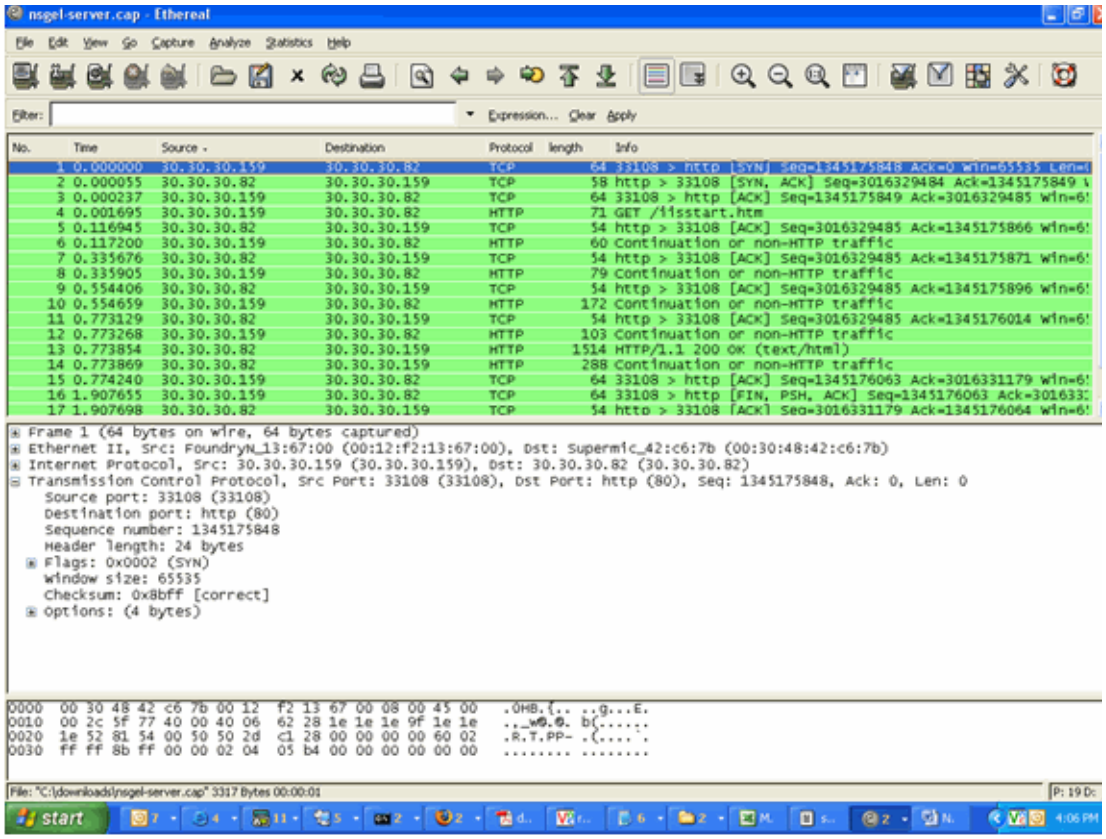


FIGURE 16 Server Capture



In these examples, the HTTP GET requests are intentionally broken down into multiple parts. In real life, you may not see GET requests divided over multiple packets.

These trace results indicate that there is degradation of performance when the ServerIronADX is configured for SSL terminate. According to the client trace, packets 1- 10 are all handshake messages and packets 11,13,15,17,19,21,23 are separate records, each having a small part of the GET request, which immediately receives an ACK.

The problem in this case is on the server side. The Microsoft 2003 server has delayed ACK ON enabled, and the delayed ACK timer is set for 200 ms. Since the Nagle Algorithm is ON by default, the ServerIronADX will not send the next packet as long as there is unacknowledged data. As shown in the server side trace, the first data that is sent to the server is the partial GET request. The complete GET request has 6 more parts. Packet number 4 is the partial GET, for which you see an ACK in packet 5.

Packet 4, length 71, (frame-size) is not a full-sized packet, so the server waits for more data packets, since it has advertised a greater window size. Because the Nagle Algorithm is enabled, the ServerIronADX does not send any more data and the server only sends an ACK after 200 ms (packet #5). The ServerIronADX waits to receive this ACK, and then sends the subsequent data packet. This process continues, and all seven packets are delayed by 200 ms resulting in total delay of 1.4 seconds, which results in the slow response time.

Resolution

There two possible approaches to this problem.

- Turn OFF delayed ACK on the server. To see how to modify or turn off delayed ACK on Windows 2003 servers, go to the following location:

  `http://support.microsoft.com/default.aspx?scid=kb;en-us;823764`

  NOTE
  This method might not be the most satisfactory, as it involves changing the registry on the servers.

- Turn OFF Nagle Algorithm on the ServerIron. Bind the TCP-profile to the port under virtual server.

The TCP Nagle Algorithm

The Nagle Algorithm was developed to address the TCP small packet problem. This problem is typically experienced when an application generates several small bytes of data at a time. As an example, one byte of user data could mean 41 bytes of packet, with 40 bytes of overhead. This situation is often referred to as the “send-side silly window” syndrome.

The Nagle Algorithm instructs the sender to buffer the data if any unacknowledged data is outstanding. Any data to be sent subsequently is held until the outstanding data is acknowledged or until there is a full packet's worth of data to send. Small amounts of data are collected by TCP and sent in a single segment.

Sometimes the Nagle Algorithm needs to be turned OFF. For example, in X-Window system, the small size messages (such as mouse movements) need to be delivered without any delay to provide real-time feedback for an interactive user.

Delayed TCP ACK

A host that is receiving a stream of TCP data segments can increase efficiency by sending fewer ACKs (acknowledgements) per data segment received using a TCP delayed ACK mechanism. A TCP should implement delayed ACKs, but no ACK should be excessively delayed.Specifically, the delay MUST be less than 0.5 seconds, and in a stream of full-sized segments there should be an ACK for at least every second segment.

The following example configures a TCP profile that turns off the delayed ACK, the Nagle Algorithm, and disables all outgoing data packets except the last one from a tcp-transmit queue. The TCP profile is then applied to Virtual Servers

Creating a TCP Profile

You can disable the following TCP features within a TCP profile: Nagle’s algorithm, the delayed ACK algorithm, and all outgoing data packets except the last one from a tcp-transmit queue. The following example creates a TCP profile named "nagleoff" within the General Configuration mode.

```
ServerIronADX(config)# tcp profile nagleoff
ServerIronADX(config-tcp-profile-nagleoff)# nagle off
ServerIronADX(config-tcp-profile-nagleoff)# delayed-ack off
ServerIronADX(config-tcp-profile-nagleoff)# push-bit off
```

Disabling Nagle's Algorithm

You can disable Nagle's algorithm within a TCP profile as shown in the following example.

```
ServerIronADX(config)# tcp profile tcpprofile1
ServerIronADX(config-tcp-profile-tcpprofile1)# nagle off
```

Syntax: [no] nagle off

Disabling the delayed ACK algorithm

You can disable the delayed ACK algorithm within a TCP profile as shown in the following example.

```
ServerIronADX(config)# tcp profile tcpprofile1
ServerIronADX(config-tcp-profile-tcpprofile1)# delayed-ack off
```

Syntax: [no] delayed-ack off

Disabling PUSH flag in outgoing data packets from the TCP transmit queue

You can disable setting PUSH flag in all outgoing data packets except when emptying the TCP transmit queue, as shown in the following example.

```
ServerIronADX(config)# tcp profile tcpprofile1
ServerIronADX(config-tcp-profile-tcpprofile1)# push-bit off
```

Syntax: [no] push-bit off

Modifying TCP receive queue size

You can modify the TCP receive queue size, as shown in the following example. The default value is 28000 bytes.

```
ServerIronADX(config)# tcp profile tcpprofile1
ServerIronADX(config-tcp-profile-tcpprofile1)# rxbuf-size <size-in-bytes>
```

Syntax: [no] rxbuf-size <*size-in-bytes*>

Modifying TCP transmit queue size

You can modify the TCP transmit queue size, as shown in the following example. The default value is 20000 bytes.

```
ServerIronADX(config)# tcp profile tcpprofile1
ServerIronADX(config-tcp-profile-tcpprofile1)# txbuf-size <size-in-bytes>
```

Syntax: [no] txbuf-size <*size-in-bytes*>

Applying the TCP profile to VIP for SSL terminate

In a SSL Terminate configuration, the TCP profile can be applied either to the **port ssl ssl-terminate** command in the Virtual Server configuration, or to the SSL profile that is being applied to the Virtual Server.

In the following example, the TCP profile is applied to the **port ssl ssl-terminate** command in the Virtual Server configuration

```
ServerIronADX(config)# server virtual-name-or-ip vip1
ServerIronADX(config-vs-vip1)# port ssl ssl-terminate sslprofile myprofile
```

Syntax: [no] port ssl ssl-terminate <*ssl-proxy*> **[tcp-proxy]**

You can also apply the TCP profile to the SSL profile. In the following example, the TCP profile "nagleoff" is applied to the SSL profile: "myprofile" and then "myprofile" is applied to the **port ssl ssl-terminate** command in

```
ServerIronADX(config)# ssl profile myprofile
ServerIronADX(config-ssl-profile-myprofile)# tcp-profile nagleoff
ServerIronADX(config-ssl-profile-myprofile)# exit
ServerIronADX(config)# server virtual-name-or-ip vip1
ServerIronADX(config-vs-vip1)# port ssl ssl-terminate sslprofile myprofile
```

Applying the TCP profile to VIP for SSL Proxy

In a SSL Proxy configuration, the TCP profile must be applied to the client and server SSL profiles that are being applied to the Virtual Server.

```
ServerIronADX(config)# server virtual-name-or-ip vip1
ServerIronADX(config-vs-vip1)# port ssl ssl-proxy clientprofile serverprofile
ServerIronADX(config)# ssl profile clientprofile
ServerIronADX(config-ssl-profile-clientprofile)# tcp-profile nagleoff
ServerIronADX(config-ssl-profile-clientprofil)# exit
ServerIronADX(config)# ssl profile serverprofile
ServerIronADX(config-ssl-profile-serverprofile)# tcp-profile nagleoff
ServerIronADX(config-ssl-profile-serverprofile)# exit
ServerIronADX(config)# server virtual-name-or-ip vip1
ServerIronADX(config-vs-vip1)# port ssl ssl-proxy clientprofile serverprofile
```

Inserting a certificate in an HTTP header

The ServerIron ADX optionally inserts the client certificate as the HTTP header, to allow the real server to access the client certificate information.

- When configuring this feature, you need to do the following in addition to a normal SSL Terminate configuration:
- Create a CSW policy to enable client certificate insertion
- Bind CSW and the CSW policy to the SSL port on the Virtual Server
- Define the Client Insertion mode and prefix within a CSW policy (optional)

Configuring a CSW Policy to enable client certificate insertion

A CSW Policy needs to be created that enables client certificate insertion. It can be configured as either a default command within a CSW policy (as shown in the following example) or as an action in response to a match in a CSW rule.

```
ServerIronADX(config)# csw-policy cswp1
ServerIronADX(config-csw-cswp1)# default rewrite request-insert client-cert
```

Syntax: [no] default rewrite request-insert client-cert

Syntax: [no] match <*csw rule name*> **rewrite request-insert client-cert**

Bind CSW and CSW policy to the Real Server

```
ServerIronADX(config)# server virtual-name-or-ip vip1
ServerIronADX(config-vs-vip1)# port ssl csw-policy "cswp1"
ServerIronADX(config-vs-vip1)# port ssl csw
```

Define client linsertion mode and prefix

The client certificate insertion mode and prefix can be optionally configured within a CSW policy as described in the following. To configure the client insertion mode, use the **default rewrite request-insert** command as shown.

```
ServerIronADX(config)# csw-policy cswp1
ServerIronADX(config-csw-cswp1)# default rewrite request-insert client-cert
```

Syntax: [no] default rewrite request-insert client-cert [entire-chain | leaf-cert | wellknown-fields]

Selecting the **entire-chain** parameter directs the ServerIron ADX to insert the entire chain including the leaf certificate in BASE64 encoded form. This is the default mode.

Selecting the **leaf-cert** parameter directs the ServerIron ADX to insert only the leaf certificate in BASE64 encoded form, even though the certificate chain is present.

If the **wellknown-fields** parameter is selected the important information of the client certificate is retrieved and inserted as the HTTP headers, in plain text. If this mode is chosen, the following headers are inserted: "Client-Cert-Version", "Client-Cert-Serial", "Client-Cert-Start", "Client-Cert-End", "Client-Cert-Subject", "Client-Cert-Subject-CN", "Client-Cert-SubjectAlt-CN", "Client-Cert-Issuer" and "Client-Cert-Issuer-CN".

You can add a prefix to the default HTTP names using the **default rewrite request-insert certheader-prefix** command. In the following example, the prefix "SSL" added to the HTTP header "Client-Cert" would become "SSL-Client-Cert".

```
ServerIronADX(config)# csw-policy cswp1
ServerIronADX(config-csw-cswp1)# default rewrite request-insert client-cert
certheader-prefix "SSL"
```

Syntax: [no] default rewrite request-insert client-cert certheader-prefix <*prefix*>

The value specified by the <prefix> variable is added to the default HTTP name.

The HTTP header names are shown in Table 18.

TABLE 18 HTTP Header Names and Descriptions

Header Names	Descriptions
Client-Cert	The entire client certificate chain or the leaf certificate.
Client-Cert-Version	Version of the client certificate.
Client-Cert-Serial	Serial number of the client certificate.
Client-Cert-Start	Date certificate not valid before.
Client-Cert-End	Date certificate not valid after.
Client-Cert-Subject	Subject's distinguished name.
Client-Cert-Subject-CN	Subject's common name.
Client-Cert-Subject-Alt-CN	Subject's alternative name.
Client-Cert-Issuer	Issuer's distinguished name.
Client-Cert-Issuer-CN	Issuer's common name.

Other protocols supported for SSL

SSL acceleration support is provided to other popular protocols such as LDAPS, POP3S, and IMAPS. Configuration of SSL acceleration support for these protocols is shown the following example.

```
ServerIronADX(config)# server real rs1
ServerIronADX(config-rs-rs1)# port pop3
ServerIronADX(config-rs-rs1)# port imap4
ServerIronADX(config-rs-rs1)# port ldap
ServerIronADX(config-rs-rs1)# exit
ServerIronADX(config)#
ServerIronADX(config)# server real rs2
ServerIronADX(config-rs-rs2)# port pop3
ServerIronADX(config-rs-rs2)# port imap4
ServerIronADX(config-rs-rs2)# port ldap
ServerIronADX(config-rs-rs2)# exit
ServerIronADX(config)#
ServerIronADX(config)# server virtual-name-or-ip vip1
ServerIronADX(config-vs-vip1)# port pop3s
ServerIronADX(config-vs-vip1)# port pop3s ssl-terminate sslprof
ServerIronADX(config-vs-vip1)# bind pop3s rs1 pop3 rs2 pop3
ServerIronADX(config-vs-vip1)# exit
ServerIronADX(config)#
ServerIronADX(config)# server virtual-name-or-ip vip1
ServerIronADX(config-vs-vip1)# port imaps
ServerIronADX(config-vs-vip1)# port imaps ssl-terminate sslprof
ServerIronADX(config-vs-vip1)# bind imaps rs1 imap4 rs2 imap4
ServerIronADX(config-vs-vip1)#
ServerIronADX(config-vs-vip1)# port ldaps
ServerIronADX(config-vs-vip1)# port ldaps ssl-terminate sslprof
ServerIronADX(config-vs-vip1)# bind ldaps rs1 ldap rs2 ldap
ServerIronADX(config-vs-vip1)# exit
ServerIronADX(config)#
```

Configuring the system max values

This section describes how to configure the following system max values on a ServerIron ADX:

- SSLv2 connection rate
- Memory limit for SSL hardware buffers
- Number of SSL profiles
- Maximum number of SSL concurrent connections

NOTE
Setting all of the system-max values to their maximum value is not advisable as it consumes a large amount of system memory. Please set these values only if necessary.

Configuring SSLv2 connection rate

You can configure the maximum connection rate for SSLv2, as shown in the following example.

```
ServerIronADX(config)# ssl-v2-rate <num-conn-per-sec>
```

Syntax: ssl-v2-rate *<num-conn-per-sec>*

The <num-conn-per-sec> variable sets the maximum connections for SSLv2 allowed per second.

Valid range: 16 to 256

Default: 100

NOTE
Please note that the connection count for the SSLv2 rate includes both client-side (Terminate / Proxy) and server-side (Proxy) connections.

Configuring memory limit for SSL hardware buffers

You can configure the maximum memory allocated for the buffers accessed by the SSL hardware, as shown in the following example.

```
ServerIronADX(config)# ssl mem-size 64
```

The <size> variable sets the maximum memory for SSL hardware buffers in MB (1048576 bytes).

Valid range: 64MB to 256MB

Default: 96MB

Configuring number of ssl profiles

You can configure the maximum number of SSL profiles, as shown in the following example:

```
ServerIronADX(config)#system-max ssl-max-profiles 64
```

Syntax: [no] system-max ssl-max-profiles <*num-max-profiles*>

The <num-max-profiles> variable sets the maximum number of SSL profiles.

Valid range: 64 to 2048 (ADX10000,4000), 64 to 1024 (ADX1000)

Default: 1024 (ADX10000,4000), 256 (ADX1000)

Configuring the maximum number of SSL concurrent connections

Use the **system-max ssl-concurrent-conn** command to set the maximum number of SSL concurrent connections, as shown in the following.

```
ServerIronADX# system-max ssl-concurrent-conn 1024
```

Syntax: [no] system-max ssl-concurrent-conn <*number-of-ssl-connections*>

The <number-of-ssl-connections> variable sets the maximum number of SSL concurrent connections.

Valid range: 1024 to 16384(ADX4000,10000), 512 to 16384 (ADX1000).

Default: 8192

SSL debug and troubleshooting commands

This section describes SSL debug and troubleshooting commands.

Diagnostics

You can run diagnostic tests on the SSL hardware devices to verify proper functionality. Please note that the diagnostic tests should not be run while SSL traffic is being processed. Also, the system should be reloaded after running the diagnostic test-suite. The diagnostic test-suite can be initiated from the MP or from individual BPs.

To run diagnostics from the MP,

```
ssl diag ServerIronADX# ssl diag <BP-slot> <BP-cpu>
<BP-slot> and <BP-cpu> refer to the BP that the diagnostic test-suite is run from

SSL chip 1: All diag tests PASSED
SSL chip 2: All diag tests PASSED
…
SSL: Diags PASSED
```

The above command runs all diagnostic tests on all SSL hardware modules, and logs whether the tests passed or failed in brief.

If additional information is needed, the diagnostic tests can be run from any BP wherein detailed information is logged on the BP console.

To run diagnostics from the BP,

SSL operations submitted to the hardware can be run in 2 modes - Blocking and Non-blocking. Blocking mode means that the CPU is polling for the result after submitting the operation to the hardware, and Non-blocking mode means that the CPU receives a callback once the operation has completed. The default mode is Blocking. To change the mode,

```
ServerIronADX1/1# ssl bp-diag mode [ blocking | non-blocking]
```

There are multiple SSL devices in the system. The default module is the first module (0). To select a specific module,

```
ServerIronADX1/1# ssl bp-diag module <SSL device ID [0...5]>
```

SSL operations submitted to the hardware can be in 2 modes - Direct and Scatter-Gather. Direct mode means that the data for any input/output variable is in one location, and Scatter-Gather mode means that the data for any input/output variable could come from multiple non-contiguous blocks. The default mode is Direct. To enable scatter-gather,

```
ServerIronADX1/1# ssl bp-diag scatter-gather [ enable | disable ]
```

```
ServerIronADX1/1# ssl bp-diag
all                                          All diagnostic tests
crypto-3des                        Crypto 3DES Test
crypto-aes                          Crypto AES Test
crypto-hmac                      Crypto HMAC Test
crypto-mod-ex                  Crypto Mod-Ex Test
crypto-rc4                           Crypto RC4 Test
key-mem                            Key Memory Test
load-ucode                         Load Microcode Test
random-num                      Random Number Generator Test
read-write-regs                 Read Write Registers Test
```

```
soft-reset                              Soft Reset Test
```

Detailed information is logged on the BP console when these tests are run.

Displaying SSL information

The following SSL Statistics information is available from the BP console within the **rconsole** mode:

- Connection proxy debug counters
- Connection proxy statistics
- Authentication statistics
- Locally stored SSL certificates
- SSL connection information
- CRL status record
- SSl debug counters
- Locally stored SSL keys
- Information about specific SSL keys
- Details of SSL profiles
- Information about SSL record sizes

Using Rconsole

To access the display command that present this information, you must enter the BP console using the **rconsole** command as shown.

```
ServerIronADX# rconsole 1 1
```

Syntax: rconsole <*slot*> <*bp*>

The <slot> variable specifies the number of the slot that the ASM is installed in whose BP console you want to access.

The <bp> variable specifies the number of the barrel processor whose BP console you want to access.

Displaying proxy debug counters

Use the **show cp debug** command in the rconsole mode to display connection proxy debug counters as shown in the following.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# show cp debug
CPU high water mark reached, SYN dropped [ 10]:   29991204
                    session alloc failed [ 36]:    1607078
            server sock error or timeout [ 84]:          8
            client sock error or timeout [ 85]:        525
```

Syntax: show cp debug

Displaying proxy statistics

Use the **show cp statistics** command in the rconsole mode to display connection proxy statistics, as shown in the following.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# show cp statistics
Client-side counters:
     SSL conn established :      24190         SSL handshake done :      17630
     TCP conn established :          0                  data rx :      17589
               rx blocked :      17589              rx unblocked :      13443
save data (await srv con) :      17259
             data tx done :      13394         no tx data to send :        730
          data tx pending :      13394           ready to tx data :      13394
          data mv pending :          0               data mv done :          0
              closed conn :      19941         remote closed conn :      20044
                 RST rcvd :          0                  RST sent :       4265
      conn close complete :      19925      sock error or timeout :          0


Server-side counters:
         conn established :      13811                  data rx :      13497
               rx blocked :      13394              rx unblocked :      13394
             data tx done :      13430         no tx data to send :          0
          data tx pending :      13408           ready to tx data :      13430
          data mv pending :          0               data mv done :          0
              closed conn :      20034         remote closed conn :      15261
                 RST rcvd :         49                  RST sent :       3500
      conn close complete :      18989      sock error or timeout :       1652
```

Syntax: show cp statistics

Displaying authentication statistics

Use the **show ssl authentication-stats** command in **rconsole** mode to display authentication statistics about certificate verification on the ServerIron ADX. This information is relevant either in the case of client certificate verification (on the client side), or while doing SSL proxy (on the server side).

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# show ssl authentication-stats
SSL certificate verification counters:
                  Success :      18384                    Failure :          0
              Unknown user :          0          Signature failed :          0
       Certificate expired :          0        Certificate revoked :          0
       Cert not yet valid  :          0      Cert signature failed :          0
 Issuer pubkey decode fail :          0           Self signed cert :      36768
     Issuer cert not found :          0    Subject Issuer mismatch :          0
     Certificate untrusted :          0        Cert chain too long :          0
     Cert not sent by peer :          0

 CRL counters:
           CRL load failed :          0       CRL signature failed :          0

             CRL not found :          0          CRL not yet valid :          0
               CRL expired :          0
```

Syntax: show ssl authentication-stats

Displaying locally stored SSL certificates

Use the **show ssl certificate** command to display locally stored SSL certificates, as shown in the following.

```
ServerIronADX# show ssl certificate *
ssl  certificate  files:

1   : cert3003.pem
2   : cert2112.pem
3   : cert2031.pem
4   : cert4030.pem
5   : cert3301.pem
6   : cert3220.pem
7   : cert2410.pem
8   : cert2014.pem
9   : cert4013.pem
10  : cert3203.pem
11  : cert3122.pem
12  : cert3041.pem
13  : cert2312.pem
14  : cert2231.pem
15  : cert2150.pem
16  : cert3501.pem
```

Syntax: show ssl certificate <*certificate-file-name*> **| ***

The <*certificate-file-name*> variable specifies the name of the certificate you want information about.

Selecting the ***** parameter displays a list of all locally stored certificates.

Displaying SSL connection information

Use the **show ssl con** command in **rconsole** mode to display SSL connection information as shown in the following.

```
ServerIronADX1/1# show ssl con
 SOCK_ID  STATE    FLAGS  SSL ptr CB_FLAGS   CP_RXQ   SSLRXQ    ERR
00000000      5 00000000 00000000 00002000 00000000 00000000      0
00000002      5 00000000 00000000 00002000 00000000 00000000      0
00000003      5 00000000 00000000 00002000 00000000 00000000      0
00000004      5 00000000 00000000 00002000 00000000 00000000      0
00000005      5 00000000 00000000 00002000 00000000 00000000      0
00000006      5 00000000 00000000 00002000 00000000 00000000      0
00000007      5 00000000 00000000 00002000 00000000 00000000      0
00000008      5 00000000 00000000 00002000 00000000 00000000      0
00000009      5 00000000 00000000 00002000 00000000 00000000      0
0000000a      5 00000000 00000000 00002000 00000000 00000000      0
0000000b      5 00000000 00000000 00002000 00000000 00000000      0
0000000c      5 00000000 00000000 00002000 00000000 00000000      0
0000000d      5 00000000 00000000 00002000 00000000 00000000      0
0000000e      5 00000000 00000000 00002000 00000000 00000000      0
0000000f      5 00000000 00000000 00002000 00000000 00000000      0
00000010      5 00000000 00000000 00002000 00000000 00000000      0
00000011      5 00000000 00000000 00002000 00000000 00000000      0
00000012      5 00000000 00000000 00002000 00000000 00000000      0
00000013      5 00000000 00000000 00002000 00000000 00000000      0
00000014      5 00000000 00000000 00002000 00000000 00000000      0
ServerIronADX 40002/7#
```

Syntax: show ssl con *

Displaying the status of a CRL record

Use the **show ssl crl** command Use the **show ssl crl** command in **rconsole** mode to display the status of a CRL record. This command shows the configuration status of the CRL, but not the contents.

```
ServerIronADX# show ssl crl verisigncrl
CRL name        : verisigncrl
IP address      : 5.1.1.5
URL             : /verisign/temp
CRL state       : Successfully Downloaded
Last downloaded : 15:30:01, Oct-31, 2005
Expiry time     : 1 hours
```

Syntax: show ssl crl <*crl-name*>

The <crl-name> variable specifies the name of the CRL record that you want to display status for.

This command shows the configuration status of the CRL, but not the contents.

```
ServerIronADX(config)# ssl crl crl1 http://192.168.5.16/temp.crl pem 1
```

```
ServerIronADX#show ssl crl <crl-name>   (on MP)
Output :
URL : /temp.crl
IP address : 192.168.5.16
CRL state : Download complete
CRL size : 2029 bytes
Expiry time : 1 hour
Next download : After 1 hour and 9 minutes
```

```
ServerIronADX3/1#show ssl crl <crl-name>   (on BP)
3/1 #sh ssl crl crl1
Certificate Revocation List (CRL):
        Version lu (0xlx)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: /C=BE/O=BELNET/OU=BEGrid/CN=BEGrid
CA/emailAddress=gridca@belnet.be
        Last Update: Oct  3 07:44:18 2005 GMT
        Next Update: Nov  7 07:44:18 2005 GMT
Revoked Certificates:
    Serial Number: 05
        Revocation Date: Dec 31 23:59:59 1999 GMT
    Serial Number: 08
        Revocation Date: Dec 31 23:59:59 1999 GMT
    Serial Number: 09
        Revocation Date: Dec 31 23:59:59 1999 GMT
    Serial Number: 0A
        Revocation Date: Dec 31 23:59:59 1999 GMT
    Serial Number: 31
        Revocation Date: Dec 31 23:59:59 1999 GMT
    Serial Number: 32
        Revocation Date: Dec 31 23:59:59 1999 GMT
    Serial Number: 33
        Revocation Date: Dec 31 23:59:59 1999 GMT
        Revocation Date: Dec 31 23:59:59 1999 GMT
        Revocation Date: Dec 31 23:59:59 1999 GMT
    Signature Algorithm: md5WithRSAEncryption
        56:dd:42:ee:3f:37:52:7a:c2:9f:92:9d:8d:84:c5:9a:4a:fc:
        43:38:b6:f1:9a:14:7f:d7:cb:6c:54:00:78:cb:9d:ac:4b:fd:
        cc:65:fe:86:5b:97:f8:40:5d:7b:16:dd:8d:91:2a:24:76:ca:
        28:e0:b1:8c:86:22:1f:94:60:67:e5:de:21:b4:77:c8:45:36:
        cf:b4:b8:2c:13:46:69:30:b3:24:b7:80:48:11:2b:47:38:a2:
        a2:50:8a:96:0c:e7:36:de:9b:eb:ee:df:d7:7c:33:a7:f1:b7:
        cc:24:eb:67:70:13:9c:c0:61:e5:85:d4:6c:61:80:b0:3a:d3:
        5f:19:cc:80:51:5d:39:19:49:b1:d9:d1:9e:ef:06:35:24:90:
        5a:b1:9b:27:0d:d0:70:a0:e2:b5:cd:a6:52:b1:9b:90:a5:3e:
        25:91:dd:b3:f9:e5:e0:f6:65:50:90:5f:64:ea:3c:00:e7:13:
        6e:f8:3a:58:1d:1f:ac:34:2b:f9:db:50:cb:93:68:fd:1d:6a:
        8f:dc:db:6e:c7:31:b8:ed:a0:5e:4c:b1:a7:65:94:40:a5:fa:
        e4:8f:97:bc:c1:c0:3f:ed:05:9a:25:3b:36:f5:3e:d8:bb:12:
        45:9f:28:4c:26:24:3d:33:72:08:ef:88:b2:d3:2a:d3:9b:1e:
```

Displaying SSL debug counters

Use the **show ssl debug** command in the rconsole mode to display debug counters, as shown in the following.

```
 ServerIronADX1/1 #show ssl debug
 Library [code]                       Description [code]:      count
     SSL [  20]         certificate verify failed [ 137]:      90219
     SSL [  20]                     uninitialized [ 301]:
```

Syntax: show ssl debug *

Displaying SSL key information

The **show ssl key** command allows you to display a list of all locally stored keys or obtain information about a specified key.

The following example displays all locally stored keys.

```
ServerIronADX# show ssl key *
ssl  key  files:
1   : key-test
2   : key1
3   : keyz
4   : keyc
5   : key7
```

Syntax: show ssl key *

The following example provides information about a specified key: "rsakey".

```
ServerIronADX# show ssl key rsakey
modulus:
    00:d6:41:66:47:98:e2:56:9d:4f:7d:e2:da:88:2e:
    eb:72:39:c9:3c:3a:be:65:73:01:a1:fc:38:c5:c0:
    bb:18:d6:65:70:ec:d5:11:57:61:2e:72:84:d4:e1:
    67:bf:87:50:50:c2:73:f3:9a:bb:41:e1:d0:d8:a0:
    d5:9a:30:15:a5:0a:7d:67:53:4a:eb:19:04:a8:82:
    72:75:74:3b:2f:d4:a5:19:09:6d:ac:1f:05:d5:c0:
    94:e5:34:93:19:f6:a8:43:7d:1b:59:44:c8:c7:6e:
    80:c2:37:d0:30:e6:66:91:ea:f3:93:88:f4:5d:29:
    c4:78:39:4e:a7:34:52:9e:63
publicExponent: lu A¹8~0xlx)
privateExponent:
    00:d4:a5:a2:32:cb:5d:51:23:de:a2:8d:c5:e1:45:
    d8:2e:cd:85:99:be:9f:fb:a6:72:67:68:22:9c:ba:
    d5:b7:28:0b:14:52:2a:82:84:9c:12:72:5c:bd:c0:
    5d:ad:2d:4a:9c:6c:f2:92:43:ef:38:cb:3b:f1:d5:
    67:4b:1a:10:4f:a5:24:c9:af:b2:5d:b3:59:68:b9:
    0b:e9:0b:e4:25:3c:d7:62:6d:e0:c3:d6:89:9f:3c:
    63:3f:f2:17:6b:e5:26:fe:26:f1:90:03:3f:3b:60:
    8b:3d:8e:c2:7a:bd:6a:78:95:3c:1b:25:82:a6:55:
    40:a1:6e:53:38:fe:2d:6b:e1
prime1:
    00:f6:6c:9b:9b:68:e5:d6:5a:eb:2b:3e:d6:45:07:
    cf:f4:ec:4a:4c:56:97:a9:76:91:b4:8c:ce:7d:02:
    b4:1c:43:9b:52:30:33:ae:29:a1:e5:97:54:d6:5e:
    d2:b9:23:40:d9:6c:1d:ee:21:7e:78:5e:63:44:14:
    a2:1b:bd:53:53
prime2:
    00:de:94:c8:08:54:42:f9:4f:f2:0b:d5:6f:71:d5:
    fc:a1:63:a2:1b:39:6e:a1:33:82:49:65:4c:27:b7:
    22:28:fc:82:1b:fc:cb:7a:fc:e6:ba:ad:41:e0:00:
    10:5d:1f:e7:25:ab:66:27:93:07:67:94:89:5f:8d:
    4b:7f:35:b6:b

    33:25:03:7e:d3:dc:b0:0a:9a:b8:95:08:1f:b7:a9:
    5f:aa:13:19:98:f7:4d:42:c0:a6:fa:7a:78:d3:b0:
    d2:14:ee:0d:b6:d7:63:14:5c:f7:ab:da:fd:cb:1c:
    6d:34:75:e7:2a:5b:63:eb:2b:b6:8b:d0:8a:76:c5:
    d6:80:0a:e5
exponent2:
    00:86:05:25:d0:c6:13:b1:94:6e:94:ab:86:38:0f:
    f5:d6:a3:6c:47:62:34:77:c1:d1:10:2a:7b:49:6a:
    9e:99:f9:38:ca:6b:53:86:11:63:48:41:ec:69:59:

    37:6b:38:47:71
coefficient:
    00:a0:e6:54:0d:34:13:8f:e3:fe:b4:7b:10:57:67:
```

Syntax: show ssl key <*keyfile-name*> **| ***

The <*keyfile-name*> variable specifies a locally stored SSL key that you want to display information for.

The * parameter displays a list of all locally stored SSL keys.

Displaying an SSL Profile

The **show ssl profile** command allows you to display the configuration of a particular SSL profile or all configured SSL profiles. The following example displays all configured SSL profiles on a ServerIron ADX.

```
ServerIronADX# show ssl profile *
SSL profile : ssl-profile-yue
Certificate file : certfile1
Key file : rsakey1
SSL cipher suite : RC4-MD5:EXP-RC4-MD5:RC4-SHA:DES-CBC-SHA:EXP-DES-CBC-SHA:DES-C
BC3-SHA:AES128-SHA:AES256-SHA:EXP1024-RC4-MD5:EXP1024-DES-CBC-SHA:EXP1024-RC4-SH
A:RC2-CBC-MD5:EXP-RC2-CBC-MD5:DES-CBC-MD5:DES-CBC3-MD5
SSL profile : 2048
Certificate file : certfile2
Key file : rsakey2
SSL cipher suite : RC4-MD5:EXP-RC4-MD5:RC4-SHA:DES-CBC-SHA:EXP-DES-CBC-SHA:DES-C
BC3-SHA:AES128-SHA:AES256-SHA:EXP1024-RC4-MD5:EXP1024-DES-CBC-SHA:EXP1024-RC4-SH
A:RC2-CBC-MD5:EXP-RC2-CBC-MD5:DES-CBC-MD5:DES-CBC3-MD5
```

Syntax: show ssl profile <*profile-name*> **| ***

The <*profile-name*> variable specifies an SSL profile that you want to display information for.

The * parameter displays all configured SSL profiles.

Displaying the session cache statistics for and SSL profile

Use the **show ssl profile session-cache stats** command to on the rconsole, as shown in the following to display the session cache statistics for a specified SSL profile.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# show ssl profile sptest session-cache stats
1024 items in the session cache
0 client connects (SSL_connect())
0 client renegotiates (SSL_connect())
0 client connects that finished
18369 server accepts (SSL_accept())
0 server renegotiates (SSL_accept())
18349 server accepts that finished
3496 session cache hits
6458 session cache misses
120 session cache timeouts
0 callback cache hits
8339 cache full overflows (ld allowed)
```

Syntax: show ssl profile <*profile-name*> **session-cache stats**

The <*profile-name*> variable specifies an SSL profile that you want to display session cache statistics for.

Displaying the certificate bound to an SSL profile

Use the **show ssl profile cert** command on the rconsole, as shown in the following, to display the certificate bound to a specified profile. This is useful when checking to see if a certificate is intact on the BPs.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# show ssl profile prof1 cert
SSL profile : prof1
Certificate:
    Data:
        Version: 00000003 (0x00000002)
        Serial Number: 00000002 (0x00000002)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=us, ST=ca, L=sj, O=brocad, OU=web, CN=www.brocade.com
        Validity
            Not Before: Jun  3 12:53:01 2009 GMT
            Not After : Jun  3 12:53:01 2010 GMT
        Subject: C=us, ST=ca, L=sj, O=brocad, OU=web, CN=www.brocade.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:d9:ef:2b:51:f9:70:b3:67:f3:17:e2:10:29:f5:
                    4d:43:8d:38:6f:bc:64:bf:92:1d:ec:4b:53:3a:3d:
                    cf:f9:35:72:55:18:b5:eb:ea:46:4a:68:2d:20:de:
                    f0:6f:05:24:89:a5:d6:a8:67:66:fe:db:5a:3d:5f:
                    8d:44:31:6f:6d:cc:17:8f:7a:a1:7b:49:8a:25:29:
                    eb:d5:5c:a9:58:20:43:f2:57:0b:7c:09:a5:8a:ac:
                    bc:71:c3:6f:58:9e:41:3f:60:8d:8d:94:a4:3c:dd:
                    1e:53:dd:9e:21:e7:c8:1e:58:e8:d5:cf:0c:41:84:
                    57:e9:9f:ad:9a:e8:16:61:f1
                Exponent: 00010001 (0x00010001)
    Signature Algorithm: sha1WithRSAEncryption
        59:0b:30:f0:06:c6:2c:b3:73:d3:40:91:5c:47:bb:d4:9c:ae:
        bb:d3:61:90:8a:d4:04:ee:81:a2:57:6e:76:a3:0b:a2:1a:08:
        9d:ce:d3:4a:95:ab:ef:62:36:6b:bc:ba:30:e3:4a:ce:1a:79:
        fe:b3:95:d6:4c:57:2d:3d:12:a9:b6:cf:aa:7d:80:83:1d:d2:
        3f:49:47:1e:d8:1f:94:47:ae:2b:f7:68:4f:51:eb:8e:30:df:
        d5:27:cd:d4:41:1a:11:85:a6:d1:1c:5b:c0:1a:60:f6:52:f0:
        8c:bd:d8:a0:56:cf:88:97:95:f4:30:e6:78:65:0e:ac:b7:80:
        fa:23
```

Syntax: show ssl profile <*profile-name*> **cert**

The <*profile-name*> variable specifies name of the profile that you want to display the certificate for.

Displaying the key that is bound to an SSL profile

Use the **show ssl profile key** command on the rconsole, as shown in the following, to display the key bound to a specified profile. This is useful when checking to see if a key is intact on the BPs.

```
ServerIronADX# rconsole 1 1
ServerIronADX# show ssl profile test1 key
SSL profile : test1
Key file : \usb0\certstor\key7.key
Private-Key: (1024 bit)
modulus:
```

```
    00:ac:6e:a1:3d:3c:0a:f3:df:e2:8d:b4:5e:d6:cb:
    90:e3:96:87:2d:bc:aa:41:64:22:fa:ea:c2:86:d8:
    b1:bc:99:c5:c6:af:87:2d:d1:2b:89:b9:31:6f:9c:
    35:03:86:9b:47:6d:82:a8:4f:88:07:dc:46:8a:87:
    86:5c:cd:15:c6:3d:de:72:05:68:0b:50:b5:77:27:
    9f:6c:33:a3:8b:2a:de:e6:f7:b3:f3:70:e6:b9:cc:
    8d:4c:84:25:b7:2f:62:d6:76:ed:93:59:87:f7:4c:
    b1:99:23:f0:9f:d9:61:d3:e1:e7:40:a0:12:6a:1d:
    f5:20:b7:2e:2b:08:9e:80:c5
publicExponent: 00010001 (0x00010001)
privateExponent:
    42:81:64:e5:16:4c:6f:25:51:df:2f:cb:48:73:39:
    4d:de:58:02:f6:fa:7f:c0:1c:91:c4:8c:04:b0:7d:
    54:ed:c6:4f:4c:92:09:c4:dc:53:01:3f:a4:f9:8d:
    a4:ef:7c:e2:7e:c5:5f:1f:55:ab:1a:75:86:a6:a0:
    d7:18:2e:a6:26:29:96:8c:e8:7e:38:df:da:5b:c5:
    90:ca:e1:3d:a3:1b:03:a7:95:e9:59:be:18:8b:dc:
    28:0a:3f:8f:a1:68:c1:07:2e:9a:8f:19:9e:e0:17:
    96:eb:7e:40:57:97:f6:13:05:e2:0e:0e:06:b8:02:
    a7:00:a3:ff:19:c2:42:9d
prime1:
    00:db:a6:28:e7:8e:ed:26:44:12:e5:bc:d5:05:98:
    d7:c2:02:f1:3c:b7:72:7e:51:7c:31:3e:9c:9a:d9:
    1a:a9:93:3c:c5:a2:27:85:1f:24:89:46:6c:4c:b8:
    bb:d0:ef:eb:d2:0e:0b:95:d5:47:bb:27:9a:50:f6:
    00:68:62:57:6b
prime2:
    00:c8:f8:09:b0:fe:87:4f:08:ab:00:f4:e7:ef:2d:
    a5:85:5a:2a:25:4f:ed:49:ba:60:55:d5:72:ce:69:
    fe:4b:ef:d7:c1:9a:a4:b3:42:68:aa:e7:9a:e0:d3:
    ee:62:99:72:df:9c:3a:1d:59:5f:74:c4:08:fe:7d:
    9a:ef:76:04:8f
exponent1:
    47:3b:bd:ec:4a:d7:f2:1f:05:99:e8:01:95:cd:19:
    bb:db:c4:6c:92:79:d9:29:88:03:58:70:e5:6f:1f:
    4c:7b:69:ac:16:88:86:8d:b1:05:ac:07:17:62:99:
    d6:8a:d8:89:c8:f0:4c:e9:5c:57:ff:e1:f5:fb:b1:
    ea:28:6a:7d
exponent2:
    00:a2:d4:b5:a5:7e:d5:4b:28:0e:c5:db:a9:00:95:
    cf:82:d8:a7:45:4c:19:4a:9f:83:e6:87:e7:59:6f:
    6f:e7:3f:11:65:80:52:ea:1b:68:8a:f0:d5:00:4d:
    36:dd:14:cf:8a:76:1e:70:21:35:c2:7a:03:7f:8f:
    6d:b5:8f:bd:e9
coefficient:
    4c:2f:16:3e:c5:b5:c9:dc:d1:68:da:8c:e2:3d:d1:
    10:48:79:80:df:a3:07:ab:84:1c:c6:86:26:f6:b2:
    42:bc:3d:65:c3:0e:d3:a0:35:0c:45:75:16:30:05:
    09:ca:a3:04:fc:26:49:c5:cf:78:87:97:ed:88:b8:
    2b:a4:00:1a
```

Syntax: show ssl profile <*profile-name*> **key**

The <profile-name> variable specifies name of the profile that you want to display the certificate for.

Displaying record size information

Use the **show ssl record-size** command in **rconsole** mode to display information regarding record size.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# show ssl record-size
                    Decrypt                                  Encrypt
Len        Count  TotalBytes   AvgBytes            Count  TotalBytes   AvgBytes
< 1k       16566    1603624          96             7413    1021277         137
< 2k           0          0  4294967295                0          0  4294967295
< 3k           0          0  4294967295                0          0  4294967295
< 4k           0          0  4294967295                0          0  4294967295
< 5k           0          0  4294967295                0          0  4294967295
< 6k           0          0  4294967295                0          0  4294967295
< 7k           0          0  4294967295                0          0  4294967295
< 8k           0          0  4294967295                0          0  4294967295
< 9k           0          0  4294967295                0          0  4294967295
<10k           0          0  4294967295                0          0  4294967295
<11k           0          0  4294967295                0          0  4294967295
<12k           0          0  4294967295                0          0  4294967295
<13k           0          0  4294967295                0          0  4294967295
<14k           0          0  4294967295                0          0  4294967295
<15k           0          0  4294967295                0          0  4294967295
<16k           0          0  4294967295                0          0  4294967295
>16k           0          0  4294967295                0          0  4294967295
```

Displaying socket information

The following socket information is available from the BP console within the **rconsole** mode.

- Socket detail in open status
- All sockets in open status
- Socket state information

To access the display command that present this information, you must enter the BP console using the **rconsole** command as shown in "Using Rconsole" on page 188.

Displaying socket details in open status

Use the **show socket id** command in the rconsole mode to display socket detail in open status, as shown in the following.

```
ServerIronADX1/1# show socket id 0
Sock       Peer        Local               State   rx_q_bytes tx_q_byptes
0000       0.0.0.0:    0 10.10.1.100:  443  LISTEN  00000000   00000000
```

Syntax: show socket id <*HEX*>

The <*HEX*> variable specifies the socket ID number in hexadecimal form of the socket that you want to display details for.

Displaying all sockets in open status

Use the **show socket list** command in the rconsole mode to display all sockets in open status, as shown in the following.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# show sock lis
Sock      Peer             Local         State    rx_q_bytes   tx_q_byptes
0000     0.0.0.0:0         10.10.6.10:     443      LISTEN       00000000  00000000
0001     0.10.6.50:3832    10.10.6.10:     443      ESTABLISHED  00000000  00000000
0002     209.157.22.1:80  10.10.6.50:3832 SYN_SENT 00000000     00000000
```

Syntax: show sock list

Displaying socket state information

Use the **show socket state** command in the rconsole mode to display socket state information, as shown in the following.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# show socket state
Socket Layer:
            Total sockets :      65000              Open sockets :      10611
              Max reached :          0        Sock mem alloc fail :          0

TCP Connection States:
                  Listen :          2                  SYN-rcvd :       1154
                SYN-sent :        353                Established :       9101
               Close-wait :          1                 FIN-wait-1 :          0
                  Closing :          0                   Last-Ack :          0
               FIN-wait-2 :          0                  Time-wait :          0

ConnProxy (client) states:
              Total count :      48000                 Used count :       4916
             Wait connect :          0        Wait connect complt :          0
               Wait reuse :          0                  Wait data :       1023
                   In Use :       3893                 Wait close :          0
        Wait remote close :          0          Wait close complt :          0
                Wait free :          0                   RST rcvd :          0

ConnProxy (server) states:
              Total count :      48000                 Used count :       4539
             Wait connect :          0        Wait connect complt :        353
               Wait reuse :          0                  Wait data :        547
                   In Use :       3638                 Wait close :          1
        Wait remote close :          0          Wait close complt :          0
                Wait free :          0                   RST rcvd :          0
Reuse pool head :           0
```

Syntax: show socket state

Displaying SSL Statistics information

The following SSL Statistics information is available from the BP console within the **rconsole** mode:

- SSL Statistics alert information
- Decoded status counters of SSL alerts
- SSL decoded client site status counters
- SSL statistical counters
- SSL crypto engine status counters

To access the display command that present this information, you must enter the BP console using the **rconsole** command as shown in "Using Rconsole" on page 188.

Displaying SSL Statistics alert information

Use the **show ssl statistics alert** command in **rconsole** mode to display decoded status counters of SSL alerts as shown in the following.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# show ssl statistics alert

SSL alert detail counters:            RX          TX
*********************************************************************
               close_notify:        7410           0
         unexpected_message:           0        2402
             bad_record_mac:           0           0
          decryption_failed:           0           0
            record_overflow:           0           0
      decompression_failure:           0           0
          handshake_failure:           0           0
             no_certificate:           0           0
            bad_certificate:           0           0
    unsupported_certificate:           0           0
        certificate_revoked:           0           0
        certificate_expired:           0           0
        certificate_unknown:           0           0
          illegal_parameter:           0           0
                 unknown_ca:           0           0
              access_denied:           0           0
               decode_error:           0           0
              decrypt_error:           0           0
        export_restrictionr:           0           0
           protocol_version:           0           0
      insufficient_security:           0           0
             internal_error:           0           0
             user_cancelled:           0           0
          no_renegotiation:           0           0
```

Syntax: show ssl statistics alert

Displaying SSL decoded client site status counters

Use the **show ssl statistics client** command in rconsole mode to display SSL decoded client site status counters as shown.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# show ssl statistics client
SSL Client statistics:
**********************************************************************
       SSL Connect attempts:   2627919          SSL Connect failed:         0
      SSL Client Hello rcvd:    746120     SSL Client Hello failed:         0
  SSL Client Key Xchng rcvd:    717585 SSL Client Key Xchng failed:         0
     SSL Client Finish rcvd:    717585    SSL Client Finish failed:         0
              Client auth ok:         0          Client auth failed:         0
  SSL Session Reuse Attempt:         0    SSL Session Reuse failed:         0
     SSL Handshake complete:    717585          SSL Connect closed:   4527817
            SSL Close count:   2617385        SSL Remote Close cnt:         4
            SSL Reset count:   2617260        SSL Remote Reset cnt:        17
    Level 1 Alerts Received:       217     Level 2 Alerts Received:         0
        Level 1 Alerts Send:       108         Level 2 Alerts Send:         0
      SSL Rx bytes from TCP: 206550026         SSL Rx bytes to TCP:1218384265
  SSL Rx Appl Data from TCP:  74514386    SSL Tx Appl Data  to TCP: 577601440
SSL Rx nonApplData from TCP: 132035640 SSL Tx nonAppl Data  to TCP: 640782825
  RSA Private Decrypt calls:         0    RSA Public Decrypt calls:         0
  RSA Private Encrypt calls:         0    RSA Public Encrypt calls:         0
       DH Compute key calls:         0       DH Generate key calls:         0
           DSA Verify calls:         0             DSA Sign calls:         0
         MD5 Raw hash calls:         0         SHA1 Raw hash calls:         0
                 3DES calls:         0                 RC4 calls:  29183053
          SSL MAC MD5 calls:  29183053          SSL MAC SHA1 calls:         0
            TLS MAC MD5 calls:          0            TLS MAC SHA1 calls:
```

Syntax: show ssl statistics client

Displaying SSL Statistics counters

Use the **show ssl statistics counters** command in rconsole mode to display SSL statistical counters as shown.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# show ssl statistics counter
SSL debug counters:
     SSL connect attempts :      16024     SSL Handshake complete :       9384
          SSL close count :      13697        SSL RemoteClose cnt :      10693
          SSL reset count :       2648       SSL Remote Reset cnt :          0
         SSL tx shutdown  :      11049              TCP close cnt :      16024
         SSL current con  :          0         SSL clientside con :          0
             SSL_Recv cnt :       9119              SSL Send cnt :       7413
          Alert(2) Rx cnt :          0           Alert(2) Tx Cnt :          0
      SSL Session Attempt :          0   SSL Session Reuse failed :          0
         SSL Rx Block cnt :      37791         SSL Rx unBlock cnt :      43356
          DMA_SSL_MONITOR :         32

       Cavium Inst err cnt :          0     Cavium reqid not found :          0
         Cavium Rd/Wr same :          0         Cavium pending cnt :          0
SSL Get Client Hello fail :          0
     FPGA ssl mon req cnt :          0      FPGA ssl mon done cnt :          0
 SSL V2 rate limit drops  :          0   Cavium Inst overflow err :          0
 SSL Alert Incorrect len  :          0    SSL Invalid record type :          0
 SSL Record with no data  :          0  SSL Insufficient data in  :          0
SSL HS MSGS in data xfer  :          0


SSL Random counters:
     Num ssl rand buffers :          2           Rand buffer size :      32760
            Current fetch :          0                Rand errors :          0


SSL debug counters:
  SSL Recv CB unblk error :          0      SSL Recv Pkt drop cnt :          0
     SSL Invalid sd error :          0      SSL ZeroCopy Recv Err :          0
       SSL read error cnt :          0          SSL recv err code :          0
     SSL read alert error :          0     SSL renegotiate start :          0
     SSL read alert error :          0         SSL max. frag. err :          0
```

Syntax: show statistics counters

Displaying SSL crypto engine status counters

Use the **show ssl statistics crypto** command in rconsole mode to display SSL crypto engine status counters as shown.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# show ssl statistics crypto
SSL crypto statistics:
************************************************************************
             Csp1Handshake:         0          Csp1HandshakeStart:         0
        Csp1HandshakeUpdate:         0         Csp1HandshakeFinish:         0
       Csp1RsaServerFullRc4:    717850  Csp1RsaServerFullRc4Finish:         0
     Csp1RsaServerVerifyRc4:         0 Csp1RsaServerVerifyRc4Finis:         0
      Csp1RsaServerFull3Des:         0 Csp1RsaServerFull3DesFinish:         0
    Csp1RsaServerVerify3Des:         0 Csp1RsaServerVerify3DesFini:         0
       Csp1RsaServerFullAes:         0  Csp1RsaServerFullAesFinish:         0
     Csp1RsaServerVerifyAes:         0 Csp1RsaServerVerifyAesFinis:         0
           Csp1OtherFullRc4:         0      Csp1OtherFullRc4Finish:         0
         Csp1OtherVerifyRc4:         0    Csp1OtherVerifyRc4Finish:         0
          Csp1OtherFull3Des:         0     Csp1OtherFull3DesFinish:         0
        Csp1OtherVerify3Des:         0   Csp1OtherVerify3DesFinish:         0
           Csp1OtherFullAes:         0      Csp1OtherFullAesFinish:         0
         Csp1OtherVerifyAes:         0    Csp1OtherVerifyAesFinish:         0
       Csp1FinishedRc4Finish:         0      Csp1Finished3DesFinish:         0
       Csp1FinishedAesFinish:         0               Csp1ResumeRc4:         0
         Csp1ResumeRc4Finish:         0             Csp1Resume3Des:         0
        Csp1Resume3DesFinish:         0              Csp1ResumeAes:         0
         Csp1ResumeAesFinish:         0        Csp1EncryptRecordRc4:  28495624
        Csp1DecryptRecordRc4:    874497       Csp1EncryptRecord3Des:         0
       Csp1DecryptRecord3Des:         0 Csp1DecryptRecord3DesRecove:         0
        Csp1EncryptRecordAes:         0        Csp1DecryptRecordAes:         0
Csp1DecryptRecordAesRecover:         0   Csp1RsaSsl20ServerFullRc4:         0
Csp1RsaSsl20ServerClientAut:         0          Csp1Ssl20ResumeRc4:         0
Csp1Ssl20ResumeClientAuthRc:         0  Csp1RsaSsl20ServerFull3Des:         0
Csp1RsaSsl20ServerClientAut:         0         Csp1Ssl20Resume3Des:         0
Csp1Ssl20ResumeClientAuth3D:         0   Csp1Ssl20DecryptRecordRc4:         0
  Csp1Ssl20EncryptRecordRc4:         0  Csp1Ssl20DecryptRecord3Des:         0
 Csp1Ssl20EncryptRecord3Des:         0                  Csp1Random:   2651352
           Csp1AllocKeyMem:         2              Csp1FreeKeyMem:         0
              Csp1StoreKey:         2              Csp1FreeKeyMem:         0
           Csp1Pkcs1v15Enc:         0          Csp1Pkcs1v15CrtEnc:         0
           Csp1Pkcs1v15Dec:         0          Csp1Pkcs1v15CrtDec:         0
           Csp1Pkcs1v15Dec:         0          Csp1Pkcs1v15CrtDec:         0
             sdram2 to dpram:    4642933              dpram to sdram2:  1552042
```

Syntax: show ssl statistics crypto

Displaying TCP IP information

The following TCP IP information is available from the BP console within the rconsole mode:

- SSL, TCP, and IP buffer information
- TCP and IP chain length statistics
- SSL, TCP, and IP queues
- SSL memory

To access the display command that present this information, you must enter the BP console using the **rconsole** command.

Displaying SSL, TCP, and IP buffer information

Use the **show tcp-ip buffers** command in rconsole mode to display SSL, TCP, and IP buffer information as shown:

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# show tcp-ip buffer
              Total          Free        Allocs         Frees      MaxInUse
  32B        414696        146884       2045939       1778127        414696
  64B         30720         14332         67150         50762         25076
 128B        135168        107922        846286        819040        117910
 256B         69632         58092        247866        236326         54982
 512B          8192          6173         29725         27706          2639
  1KB         16384         16384        129930        129930          8608
  2KB         24576         24576         77162         77162          8607
  8KB          8192          8191        285319        285318          6195
8.5KB          2048             0          2048             0          2048
 14KB           192           192             0             0             0

 SSL buffers allocated from pcie memory space ; size = 213509280 bytes
```

Syntax: show tcp-ip buffers

Displaying TCP, and IP chain length statistics

Use the **show tcp-ip chain-statistics** command in rconsole mode to display TCP and IP chain length statistics as shown.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# show tcp-ip chain-statistics
                  TCP->App                                App->TCP
Len       Count  TotalBytes   AvgBytes           Count  TotalBytes   AvgBytes
  1      139369   35133603         252           19775    1265600         64
  2           0          0  4294967295           76315   10870122        142
  3           0          0  4294967295               0          0  4294967295
  4           0          0  4294967295           17630    1181210         67
  5           0          0  4294967295               0          0  4294967295
  6           0          0  4294967295           24190   26078654       1078
  7           0          0  4294967295               0          0  4294967295
  8           0          0  4294967295               0          0  4294967295
  9           0          0  4294967295               0          0  4294967295
 10           0          0  4294967295               0          0  4294967295
 11           0          0  4294967295               0          0  4294967295
 12           0          0  4294967295               0          0  4294967295
 13           0          0  4294967295               0          0  4294967295
 14           0          0  4294967295               0          0  4294967295
 15           0          0  4294967295               0          0  4294967295
 16           0          0  4294967295               0          0  4294967295
>16           0          0  4294967295               0          0  4294967295
ServerIronADX 40002/7#
```

Syntax: show tcp-ip chain-statistics

Displaying TCP and IP statistics

Use the **show tcp-ip statistics** command in rconsole mode to display TCP and IP statistics as shown in the following.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# show tcp-ip statistics
Driver Layer:
                  Rx cnt :     405603         Pkt mem alloc fail :          0
                  Tx cnt :     445200              No tx buffers :          0
    Tx frame data too big :          0            Tx hdrs too long :          0
           Tx Seg no data :          0              Tx seg w/ data :          0
           Tx Seg chunked :          0

IP (rx) Layer:
                  Rx Cnt :     405603         Fwd to upper layer :     405603
              Cksum error :          0                Bad options :          0
             Bad protocol :          0               Pkt truncated :          0
               Bad header :          0
                  Rx frag :          0             Frag reassm ok :          0
      Frag tmr alloc fail :          0        Frag mem alloc fail :          0
     Frag max num reached :          0      Frag max size reached :          0
         Frag dup data rx :          0

IP (tx) Layer:
      Rx from upper layer :     447386               dest unreach :       2186

TCP (rx) Layer:
                  Rx cnt :     405603           Rx bad ctl flags :          0
                  Rx SYN :      24190                 Rx SYN-ACK :      19793
                  Rx FIN :      35310                     Rx RST :         62
     Duplicate Seg (full) :       5161       Duplicate Seg (part) :          0
 Rx seg out of wnd (full) :          0   Rx seg out of wnd (part) :          0
Cksum error (HW verified) :          0  Cksum error (SW verified) :          0
              Bad hdr len :          0        Bad data truncation :          0
       PAWS bad timestamp :          0          Rx zero wnd probe :          0
             Rx Q corrupt :          0        Rx unacceptable seg :          7

TCP (tx) Layer:
      Rx from upper layer :     137910         Retransmitting seg :      24766
                  Tx SYN :      47562                 Tx SYN-ACK :      25460
                  Tx FIN :      38338                     Tx RST :       9194
           Bad socket num :          0            Bad data buffer :          0
             Bad data len :          0              Sock send err :          0
      Sock already closed :          0       Connection not estab :          0
             Data too big :          0                  Tx Q full :          0
                  Tx busy :          0            Send wnd shrunk :        896
             Tx Q corrupt :          0          Tx zero wnd probe :          0
```

Syntax: show tcp-ip statistics

Show SSL memory

Use the **show ssl mem** command in rconsole mode to display SSL memory statistics as shown in the following.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# show ssl mem
Total SSL Buffer Usage:
Size: 32B    64B   128B   256B  512B  1K   2K   8K   8.5K   14K
Free  23dc4  037fc 1a592  e2ec  181d  4000 6000 1fff 0000 00c0
Used  41624  04004 06a6e  2d14  07e3  0000 0000 0001 0800 0000


SSL buffers allocated from pcie memory space ; size = 213509280 bytes

SSL buffer dyn allocations (non-std size chunks) : 197195
SSL buffer dyn allocations (few std size chunks depleted) 1126328
SSL buffer dyn frees : 11208
SSL buffer malloc fail cnt  : 0


Total Used DP Buffer cnt: 0x00000000
Total Used DP Buffer size: 0x00000000


Max Used DP Buffer: 0x00000000
ServerIronADX 40002/7#
```

Syntax: show ssl mem

ASM SSL dump commands

The following ASM SSL dump commands can be used for troubleshooting your ServerIron ADX system. Because these commands are performance intensive, use discretion when using them within a production system.

asm dm ssldump

Use the **asm dm ssldump** command on the BP to display all transmit and receive SSL packets.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# asm dm ssldump
SSL transmit and receive packets in on now
2/1 #     1     135834ms 10.10.1.101:443->10.10.2.66:51632  Application Data
     2     135835ms 10.10.1.101:443->10.10.2.189:15664  Application Data
     3     135835ms 10.10.1.101:443->10.10.2.98:16578  Application Data
     4     135835ms 10.10.1.101:443->10.10.2.158:10554  Application Data
     5     135835ms 10.10.1.101:443->10.10.2.127:10650  Application Data
     6     135836ms 10.10.1.101:443->10.10.2.194:16461  Application Data
     7     135836ms 10.10.1.101:443->10.10.2.158:10601  Application Data
     8     135836ms 10.10.2.33:46947->10.10.1.101:443  Application Data
     9     135836ms 10.10.1.101:443->10.10.2.127:9521  Application Data
    10     135836ms 10.10.1.101:443->10.10.2.33:46933  Application Data
    11     135837ms 10.10.1.101:443->10.10.2.189:15664  Application Data
    12     135837ms 10.10.1.101:443->10.10.2.66:51632  Application Data
    13     135837ms 10.10.1.101:443->10.10.2.127:10650  Application Data
    14     135838ms 10.10.1.101:443->10.10.2.98:16578  Application Data
    15     135838ms 10.10.1.101:443->10.10.2.158:10554  Application Data
    16     135838ms 10.10.1.101:443->10.10.2.127:9521  Application Data
    17     135838ms 10.10.1.101:443->10.10.2.33:46947  Application Data
    18     135838ms 10.10.1.101:443->10.10.2.194:16461  Application Data
    19     135839ms 10.10.1.101:443->10.10.2.158:10601  Application Data
    20     135839ms 10.10.1.101:443->10.10.2.33:46933  Application Data
    21     135839ms 10.10.1.101:443->10.10.2.33:46947  Application Data
    22     135840ms 10.10.1.101:443->10.10.2.33:46947  Application Data
    23     135840ms 10.10.1.101:443->10.10.2.66:51632  Application Data
    24     135840ms 10.10.1.101:443->10.10.2.189:15664  Application Data
    25     135840ms 10.10.1.101:443->10.10.2.127:10650  Application Data
    26     135841ms 10.10.1.101:443->10.10.2.98:16578  Application Data
    27     135841ms 10.10.1.101:443->10.10.2.158:10554  Application Data
    28     135841ms 10.10.1.101:443->10.10.2.127:9521  Application Data
    29     135841ms 10.10.1.101:443->10.10.2.194:16461  Application Data
    30     135841ms 10.10.1.101:443->10.10.2.33:46933  Application Data
    31     135841ms 10.10.1.101:443->10.10.2.158:10601  Application Data
    32     135842ms 10.10.1.101:443->10.10.2.33:46947  Application Data
    33     135842ms 10.10.1.101:443->10.10.2.66:51632  Application Data
    34     135843ms 10.10.1.101:443->10.10.2.189:15664  Application Data
    35     135843ms 10.10.1.101:443->10.10.2.127:10650  Application Data
    36     135843ms 10.10.1.101:443->10.10.2.98:16578  Application Data
    37     135844ms 10.10.1.101:443->10.10.2.158:10554  Application Data
    38     135844ms 10.10.1.101:443->10.10.2.194:16461  Application Data
    39     135844ms 10.10.1.101:443->10.10.2.127:9521  Application Data
    40     135844ms 10.10.1.101:443->10.10.2.158:10601  Application Data
    41     135844ms 10.10.1.101:443->10.10.2.33:46933  Application Data
    42     135845ms 10.10.1.101:443->10.10.2.33:46947  Application Data
    43     135845ms 10.10.1.101:443->10.10.2.66:51632  Application Data
    44     135846ms 10.10.1.101:443->10.10.2.189:15664  Application Data
```

Syntax: asm dm ssldump

asm dm ssldump both

Use the **asm dm ssldump both** command on the BP to display both client and server SSL packets.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# asm dm ssldump both
Debug both client and server packets
```

Syntax: asm dm ssldump both

asm dm ssldump client

Use the **asm dm ssldump client** command on the BP to display client SSL packets only.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# asm dm ssldump client
Debug client packets only
```

Syntax: asm dm ssldump client

asm dm ssldump filter

Use the **asm dm ssldump filter** command on the BP to set up SSL packet filters.

Use the **asm dm ssldump filter** command with the following options to display SSL packets with matching destination IP addresses.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# asm dm ssldump filter 1 dpa <ip_addr>
```

Syntax: asm dm ssldump filter <*num*> **dpa** <*ip_addr*>

Use the **asm dm ssldump filter** command with the following options to display SSL packets with matching source IP addresses.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# asm dm ssldump filter 1 spa 1<ip_addr>
```

Syntax: asm dm ssldump filter <*num*> **spa** <*ip_addr*>

Use the **asm dm ssldump filter** command with the following options to display SSL packets with matching destination ports, use the following syntax:

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1#asm dm ssldump filter 1 dport 80
```

Syntax: asm dm ssldump filter <*num*> **dport** <*num*>

Use the **asm dm ssldump filter** command with the following options to display SSL packets with matching source ports, use the following syntax:

```
ServerIronADX# asm dm ssldump filter 1 sport 80
```

Syntax: asm dm ssldump filter <*num*> *sport* <*num*>

asm dm ssldump mode brief

Use the **asm dm ssldump mode brief** command on the BP to display SSL packet summary information.

asm dm ssldump mode detail

Use the **asm dm ssldump mode detail** command on the BP to display SSL handshake packet detail information.

asm dm ssldump mode decrypt

Use the **asm dm ssldump mode decrypt** command on the BP to display SSL decrypted received packets only.

asm dm ssldump receive

Use the **asm dm ssldump receive** command on the BP to display received packets only.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# asm dm ssldump receive

1    0 ms 10.10.6.50:3633->10.10.6.10:443  Handshake ClientHello
2  1906ms 10.10.6.50:3633->10.10.6.10:443  Handshake ClientKeyEx
3  1935ms 10.10.6.50:3633->10.10.6.10:443  Handshake ChangeCipherSpec
4  1976ms 10.10.6.50:3633->10.10.6.10:443  Handshake Finished
5  4709ms 10.10.6.50:3633->10.10.6.10:443  Application Data
```

asm dm ssldump send

Use the **asm dm ssldump send** command on the BP to display sent packets only.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# asm dm ssldump send
6  158955ms 10.10.6.10:443->10.10.6.50:3645  Handshake ServerHello
7  159064ms 10.10.6.10:443->10.10.6.50:3645  Handshake ServerCert
8  159105ms 10.10.6.10:443->10.10.6.50:3645  Handshake ServerHelloDone
9  162812ms 10.10.6.10:443->10.10.6.50:3645  Handshake ChangeCipherSpec
10 162837ms 10.10.6.10:443->10.10.6.50:3645  Handshake Finished
11 164822ms 10.10.6.10:443->10.10.6.50:3645  Application Data
```

asm dm ssldump server

asm dm ssldump server

Use the **asm dm ssldump server** command on the BP to display client SSL packets only.

```
ServerIronADX# rconsole 1 1
ServerIronADX1/1# asm dm ssldump server
Debug server packets only
ServerIronADX4/1#
```

Syntax: asm dm ssldump server

asm dm ssldump no-limit

Use the **asm dm ssldump no-limit** command to keep dumping all captured SSL packets until the dump is stopped by entering "asm dm ssldump"

Syntax: asm dm ssldump max <*count*>

asm dm ssldump max

Use the **asm dm ssldump max** <*count*> command to limit the number of packets logged on the console.

Syntax: asm dm ssldump max <*count*>

The default value is 50.